

Linux Perf Tools

Probe & Trace

Arnaldo Carvalho de Melo

Red Hat Inc.

February 6, 2015

- probe & trace
- Alternatives to existing tools
- But with new features!
- Using (relatively) recent kernel infrastructures
- perf, ftrace, kprobes, uprobes
- More coming: eBPF

- Strengths
- Limitations
- Missing features
- Current and Future work

- Tool output examples
- As non root user as much as possible

- strace
- no ptrace: lower overhead
- doesn't stop the target
- Syscall argument prettifier
- More targets: syswide, cpu, others
- record/report workflow

strace overhead

```
# dd if=/dev/zero of=/dev/null bs=1 count=500k  
512000+0 records in  
512000+0 records out  
512000 bytes (512 kB) copied, 0.186082 s, 2.8 MB/s  
#
```

strace overhead

```
# strace -eaccept dd if=/dev/zero of=/dev/null bs=1 count=500k
512000+0 records in
512000+0 records out
512000 bytes (512 kB) copied, 32.2895 s, 15.9 kB/s
#
```

- 173x times slower
- Worst case, dd emits syscalls as fast as it can
- Brendan Gregg's article mentions 442x slow down, same example
- ptrace pauses the application twice
- syscall enter and exit
- Stops at each syscall, even with just accept selected
- www.brendangregg.com/blog/2014-05-11/strace-wow-much-syscall.html

perf trace overhead

```
# perf trace -eaccept dd if=/dev/zero of=/dev/null bs=1 count=500k
512000+0 records in
512000+0 records out
512000 bytes (512 kB) copied, 2.2436 s, 228 kB/s
#
```

- 14x times slower
- Again worst case, dd emits syscalls as fast as it can
- No workload stop at syscall enter/exit
- Collects info at syscall enter and exit
- Further optimizations possible
- Currently uses raw_syscalls:sys_{enter,exit} tracepoints
- Registers each syscall, even just accept being selected
- But in this example could have used just syscalls:sys_{enter,exit}_accept

- Multiple mechanisms used
- Debugfs, tracepoints have different knobs to deal with
- procfs knobs

```
# ls /proc/sys/kernel/perf_event_*  
/proc/sys/kernel/perf_event_max_sample_rate  
/proc/sys/kernel/perf_event_mlock_kb  
/proc/sys/kernel/perf_event_paranoid  
# cat /proc/sys/kernel/perf_event_max_sample_rate  
50000  
# cat /proc/sys/kernel/perf_event_mlock_kb  
516  
# cat /proc/sys/kernel/perf_event_paranoid  
1
```

perf_event_paranoid

```
$ perf top --stdio
```

```
Error:
```

```
You may not have permission to collect system-wide stats.
```

```
Consider tweaking /proc/sys/kernel/perf_event_paranoid:
```

```
-1 - Not paranoid at all
```

```
0 - Disallow raw tracepoint access for unpriv
```

```
1 - Disallow cpu events for unpriv
```

```
2 - Disallow kernel profiling for unpriv
```

```
$
```

Permissions

```
$ ps ax|grep git
 3328 pts/0    S+      0:00 git remote update clark
 3329 pts/0    S+      0:00 git fetch --multiple clark
 3330 pts/0    S+      0:04 git fetch --append clark
 3370 pts/1    S+      0:00 grep git
$ trace -p 3330
Error: Unable to find debugfs
Hint: Was your kernel compiled with debugfs support?
Hint: Is the debugfs filesystem mounted?
Hint: Try 'sudo mount -t debugfs nodev /sys/kernel/debug'
```

Permissions

```
$ sudo mount -t debugfs nodev /sys/kernel/debug
$ trace -p 3330
Error: No permissions to read /sys/kernel/debug/tracing/events/raw_syscalls/*
Hint: Try 'sudo mount -o remount,mode=755 /sys/kernel/debug'
```

Permissions

```
$ sudo mount -o remount,mode=755 /sys/kernel/debug
$ trace -p 3330
  0.012 ( 0.006 ms): read(fd: 30<socket:[849]>, buf: 0x7215a0, count: 4      ) = 4
  0.070 ( 0.017 ms): write(fd: 31<socket:[849]>, buf: 0x11e12c0, count: 1604 ) = 1604
267.908 (267.836 ms): read(fd: 30<socket:[849]>, buf: 0x7fff503efb20, count: 4) = 4
267.914 ( 0.004 ms): read(fd: 30<socket:[849]>, buf: 0x7215a0, count: 4      ) = 4
267.945 ( 0.011 ms): write(fd: 31<socket:[849]>, buf: 0x11e12c0, count: 1604 ) = 1604
533.696 (265.749 ms): read(fd: 30<socket:[849]>, buf: 0x7fff503efb20, count: 4) = 4
533.701 ( 0.003 ms): read(fd: 30<socket:[849]>, buf: 0x7215a0, count: 4      ) = 4
533.800 ( 0.013 ms): write(fd: 31<socket:[849]>, buf: 0x11e12c0, count: 1604 ) = 1604
^C
#
```

System wide tracing for non root user

```
$ trace
Error: Operation not permitted.
Hint: Check /proc/sys/kernel/perf_event_paranoid setting.
Hint: For system wide tracing it needs to be set to -1.
Hint: Try: 'sudo sh -c "echo -1 > /proc/sys/kernel/perf_event_paranoid"'
Hint: The current value is 1.
$
```

System wide tracing for non root user

```
$ sudo sh -c "echo -1 > /proc/sys/kernel/perf_event_paranoid"
$ trace | egrep -v trace\|grep | head -10
 243.634 (0.000 ms): qpidd/382 ... [continued]: futex() = -1 ETIMEDOUT Connection timed out
 243.654 (0.004 ms): qpidd/382 futex(uaddr: 0x7cb578, op: WAKE|PRIV, val: 1) = 0
2243.629 (1999.962 ms): qpidd/382 futex(uaddr: 0x7cb5a4, op: WAIT_BITSET|PRIV|CLKRT,
                               val: 3775, utime: 0x7ffaa7a7bc90,
                               val3: 4294967295) = -1 ETIMEDOUT Connection timed out
2243.636 (0.002 ms): qpidd/382 futex(uaddr: 0x7cb578,
                               op: WAKE|PRIV, val: 1) = 0
2249.243 (0.000 ms): hald-addon-acp/875 ... [continued]: nanosleep() = 0
2249.401 (0.154 ms): hald-addon-acp/875 socket(family: LOCAL,
                               type: STREAM) = 4
2249.424 (0.020 ms): hald-addon-acp/875 connect(fd: 4, servaddr: 0x7fff8e632870,
                               addrLen: 110) = -1 ENOENT No such file or directory
2249.435 (0.002 ms): hald-addon-acp/875 close(fd: 4) = 0
2249.453 (0.002 ms): hald-addon-acp/875 rt_sigprocmask(how: BLOCK,
                               nset: 0x7fff8e6326f0,
                               oset: 0x7fff8e632670,
                               sigsetsize: 8) = 0
2249.456 (0.002 ms): hald-addon-acp/875 rt_sigaction(sig: CHLD, oact: 0x7fff8e632480,
                               sigsetsize: 8) = 0
$
```


Mixing with tracepoints

- Same as with other tools: `-event`
- Just implemented
- But `-e` already used as equivalent to `strace`

Mixing with tracepoints

```
$ trace --event sched:sched_process_exec,sched:sched_switch,sched:sched_process_exit sleep 1
 0.072 sched:sched_process_exec:filename=/usr/bin/sleep pid=12927 old_pid=12927)
 0.113 ( 0.003 ms): brk(                                ) = 0x1e20000
<SNIP>
 0.522 ( 0.002 ms): close(fd: 3                        ) = 0
 0.588 sched:sched_switch:sleep:12927 [120] S ==> kworker/1:1:11751 [120])
1001.884 (1001.301 ms): nanosleep(rqtp: 0x7fff518583b0) ) = 0
1001.972 ( 0.009 ms): close(fd: 1                      ) = 0
1001.989 ( 0.006 ms): close(fd: 2                      ) = 0
1002.001 ( 0.000 ms): exit_group(
1002.302 sched:sched_process_exit:comm=sleep pid=12927 prio=120)
$
```

No syscalls

```
$ trace --no-sys --ev sched:*process_exec,sched:*switch,sched:*process_exit sleep 1
  0.070 sched:sched_process_exec:filename=/usr/bin/sleep pid=15405 old_pid=15405)
  0.543 sched:sched_switch:sleep:15405 [120] S ==> swapper/0:0 [120])
1002.006 sched:sched_process_exit:comm=sleep pid=15405 prio=120)
$
```

- Similar to 'perf record' + 'perf report'
- Collect in one machine, analyse in another
- Needs better error handling, like 'live mode'
- Running as non root user has several issues
- Maybe I will be able to fix it in time for devconf ;-)

```
$ trace record usleep
```

```
Neither raw_syscalls nor syscalls events exist.
```

```
$ trace usleep
```

```
Error: No permissions to read /sys/kernel/debug/tracing/events/raw_s
```

```
Hint: Try 'sudo mount -o remount,mode=755 /sys/kernel/debug'
```

```
$ sudo mount -o remount,mode=755 /sys/kernel/debug
```

```
$ trace record usleep
```

```
Permission error mapping pages.
```

```
Consider increasing /proc/sys/kernel/perf_event_mlock_kb,
```

```
or try again with a smaller value of -m/--mmap_pages.
```

```
(current value: 1024)
```

```
$ trace -e nanosleep usleep
```

```
1.960 ( 0.058 ms): nanosleep(rqtp: 0x7fffd9d1e730) = 0
```

```
$
```

Argument beautifiers

- Similar to strace
- But some need more data collected in the kernel
- Use 'perf probe' to collect those!

- Create dynamic probes in arbitrary places, then activate
- Becomes a tracepoint
- Can collect variables
- Non interactive debugging
- kernel: kprobes
- Userspace: uprobes

Where to insert a probe

```
# perf probe -L getname_flags
<getname_flags@usr/src/debug/kernel-3.17.fc20/linux-3.17.8-200.fc20.x86_64/fs/namei.c:0>
    0  getname_flags(const char __user *filename, int flags, int *empty)
    1  {
        struct filename *result, *err;
        int len;
        long max;
        char *kname;

<SNIP>
    25  len = strncpy_from_user(kname, filename, max);
    26  if (unlikely(len < 0)) {
    27  err = ERR_PTR(len);
        goto error;
    }

<SNIP>
    65  result->uptr = filename;
    66  result->aname = NULL;
        audit_getname(result);
        return result;

        error:
    71  final_putname(result);
    72  return err;
    73  }
```


Source code view

- Shows line number offsets from start of the function
- Where probes can be inserted
- Requires DWARF information
- Matching debuginfo packages or 'gcc -g' built binaries

Inserting the probe

```
# perf probe 'vfs_getname=getname_flags:65 pathname=filename:string'  
Added new event:  
  probe:vfs_getname      (on getname_flags:65 with pathname=filename:string)
```

You can now use it in all perf tools, such as:

```
perf record -e probe:vfs_getname -aR sleep 1
```

```
# perf probe --list  
  probe:vfs_getname      (on getname_flags:65@fs/namei.c with pathname)
```

Trying it

```
# perf record -e probe:vfs_getname touch My-File-Name
[ perf record: Woken up 1 times to write data ]
# perf script
touch 880 [2] 0.65794: probe:vfs_getname: (ffff8120b573) pathname="/etc/ld.so.preload"
touch 880 [2] 0.65802: probe:vfs_getname: (ffff8120b573) pathname="/etc/ld.so.cache"
touch 880 [2] 0.65817: probe:vfs_getname: (ffff8120b573) pathname="/lib64/libc.so.6"
touch 880 [2] 0.66003: probe:vfs_getname: (ffff8120b573) pathname="/usr/lib/locale/locale-arc
touch 880 [2] 0.66048: probe:vfs_getname: (ffff8120b573) pathname="My-File-Name"
#
```

- trace uses this "vfs_getname" wannabe tracepoint if available
- "wannabe tracepoints" can be prototyped in this way
- Eventually some may become real tracepoints
- Changes in kernels may be isolated via a standard interface
- Location/variable name may change
- But "vfs_getname" and "pathname" remains

trace using vfs_getname

```
# trace -e open,close,dup2 touch My-File-Name | grep My
0.991 (0.002 ms): dup2(oldfd: 3<My-File-Name>) = 0
0.995 (0.001 ms): close(fd: 3<My-File-Name> ) = 0
#
```

trace needs more help from probe

- When copying syscall arguments from userspace to the kernel
- How to signal how much to copy?
- Automatically creating the probes?

probe integration with other tools

- Probes are just tracepoints
- All other features presents: callchains
- Quick scripting in perl, python

The life of a ping packet

```
$ perf probe -L icmp_rcv
<icmp_rcv@usr/src/debug/kernel-3.17.fc20/linux-3.17.8-200.fc20.x86_64/net/ipv4/icmp.c:0>
  0 int icmp_rcv(struct sk_buff *skb)
  1 {
    struct icmphdr *icmph;
  3 struct rtable *rt = skb_rtable(skb);
    struct net *net = dev_net(rt->dst.dev);
<SNIP>
 28 if (skb_checksum_simple_validate(skb))
    goto csum_error;

 31 if (!pskb_pull(skb, sizeof(*icmph)))
    goto error;

 34 icmph = icmp_hdr(skb);
<SNIP>
 51 if (rt->rt_flags & (RTCF_BROADCAST | RTCF_MULTICAST)) {
    /*
     * RFC 1122: 3.2.2.6 An ICMP_ECHO to broadcast MAY be
     *   silently ignored (we let user decide with a sysctl).
     * RFC 1122: 3.2.2.8 An ICMP_TIMESTAMP MAY be silently
     *   discarded if to broadcast/multicast.
     */
 58 if ((icmph->type == ICMP_ECHO ||
 59     icmph->type == ICMP_TIMESTAMP) &&
```


Inserting the probe

```
# perf probe icmp_rcv:59
```

```
Added new event:
```

```
  probe:icmp_rcv          (on icmp_rcv:59)
```

You can now use it in all perf tools, such as:

```
perf record -e probe:icmp_rcv -aR sleep 1
```

```
#
```

Using it

```
# perf record -e probe:icmp_rcv --call-graph dwarf ping -b 127.255.255.255
WARNING: pinging broadcast address
PING 127.255.255.255 (127.255.255.255) 56(84) bytes of data.
^C
--- 127.255.255.255 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 999ms

[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.033 MB perf.data (2 samples) ]
#
```

Callchain from kernel to userspace

```
# perf report
```

```
root@zoo:/home/acme
Samples: 2 of event 'probe:icmp_rcv', Event count (approx.): 2
Overhead Command Shared Object Symbol
- 100.00% ping [kernel.vmlinux] [k] icmp_rcv
icmp_rcv
ip_local_deliver_finish
ip_local_deliver
ip_rcv_finish
ip_rcv
__netif_receive_skb_core
__netif_receive_skb
process_backlog
net_rx_action
__do_softirq
do_softirq_own_stack
do_softirq
__local_bh_enable_ip
ip_finish_output
ip_output
ip_local_out_sk
ip_send_skb
ip_push_pending_frames
raw_sendmsg
inet_sendmsg
do_sock_sendmsg
__sys_sendmsg
__sys_sendmsg
sys_sendmsg
system_call
__sendmsg_nocancel
send_probe
pinger
main_loop
main
__libc_start_main
start
Press '?' for help on key bindings
```

Callchain from kernel to userspace - shift+V

```
# perf report
```

```
root@zoo:/home/acme
Samples: 2 of event 'probe:icmp_rcv', Event count (approx.): 2
Overhead Command Shared Object Symbol
- 100.00% ping [kernel.vmlinux] [k] icmp_rcv
  icmp_rcv [kernel.vmlinux]
  ip_local_deliver_finish [kernel.vmlinux]
  ip_local_deliver [kernel.vmlinux]
  ip_rcv_finish [kernel.vmlinux]
  ip_rcv [kernel.vmlinux]
  __netif_receive_skb_core [kernel.vmlinux]
  __netif_receive_skb [kernel.vmlinux]
  process_backlog [kernel.vmlinux]
  net_rx_action [kernel.vmlinux]
  __do_softirq [kernel.vmlinux]
  do_softirq_own_stack [kernel.vmlinux]
  do_softirq [kernel.vmlinux]
  __local_bh_enable_ip [kernel.vmlinux]
  ip_finish_output [kernel.vmlinux]
  ip_output [kernel.vmlinux]
  ip_local_out_sk [kernel.vmlinux]
  ip_send_skb [kernel.vmlinux]
  ip_push_pending_frames [kernel.vmlinux]
  raw_sendmsg [kernel.vmlinux]
  inet_sendmsg [kernel.vmlinux]
  do_sock_sendmsg [kernel.vmlinux]
  __sys_sendmsg [kernel.vmlinux]
  __sys_sendmsg [kernel.vmlinux]
  sys_sendmsg [kernel.vmlinux]
  system_call [kernel.vmlinux]
  __sendmsg_nocancel libc-2.18.so
  send_probe ping
  pinger ping
  main_loop ping
  main ping
  __libc_start_main libc-2.18.so
  start ping
Press '?' for help on key bindings
```

Using scripts with probes

```
# perf script -g python
generated Python script: perf-script.py
# cat perf-script.py
# remove some boilerplate

def probe__icmp_rcv(event_name, context, common_cpu,
                    common_secs, common_nsecs, common_pid, common_comm,
                    common_callchain, __probe_ip):

    print "__probe_ip=%u" % (__probe_ip)

    for node in common_callchain:
        if 'sym' in node:
            print "\t[%x] %s" % (node['ip'], node['sym']['name'])
        else:
            print "          [%x]" % (node['ip'])

    print "\n"
#
```

Listing probeable functions in userspace DSO

```
# perf probe -F /lib64/libc-2.12.so|grep ^m|head -10
madvise
malloc
malloc@plt
malloc_info
mblen
mbstowcs
mbtowc
mcheck
mcheck_check_all
mcheck_pedantic
#
```

Adding userspace probe

```
# perf probe -x /lib64/libc-2.12.so malloc
Added new event:
  probe_libc:malloc      (on 0x79b80)
```

You can now use it in all perf tools, such as:

```
perf record -e probe_libc:malloc -aR sleep 1
```

```
#
```

Collecting callchains with stack chunks

```
# perf record -e probe_libc:* -g dwarf,1024 sleep 2  
[ perf record: Woken up 1 times to write data ]  
[ perf record: Captured and wrote 0.058 MB perf.data (~2547  
#
```


Report snapshot

```
# cat perf.hist.5
- 100.00% sleep  libc-2.12.so  [.] malloc
  - malloc
    - 45.16% __strdup
      + 85.71% setlocale
      + 7.14% _nl_load_locale_from_archive
      + 7.14% __textdomain
    + 38.71% _nl_intern_locale_data
    + 6.45% _nl_normalize_codeset
    + 3.23% _nl_load_locale_from_archive
    - 3.23% new_composite_name
      setlocale
      0x4014ec
      __libc_start_main
      0x4011f9
    + 3.23% set_binding_values
```

#

Verbose report snapshot

```
# cat perf.hist.6
- 100.00% sleep libc-2.12.so  [.] malloc
  - malloc libc-2.12.so
    - 45.16% __strdup libc-2.12.so
      + 85.71% setlocale libc-2.12.so
      + 7.14% _nl_load_locale_from_archive libc-2.12.so
      + 7.14% __textdomain libc-2.12.so
    + 38.71% _nl_intern_locale_data libc-2.12.so
    + 6.45% _nl_normalize_codeset libc-2.12.so
    + 3.23% _nl_load_locale_from_archive libc-2.12.so
    - 3.23% new_composite_name libc-2.12.so
      setlocale libc-2.12.so
      0x4014ec sleep
      __libc_start_main libc-2.12.so
      0x4011f9 sleep
    + 3.23% set_binding_values libc-2.12.so
# rpm -qf 'which sleep'
coreutils-8.4-19.el6.x86_64
# rpm -q coreutils-debuginfo
package coreutils-debuginfo is not installed
# rpm -q glibc-debuginfo
glibc-debuginfo-2.12-1.80.el6_3.4.x86_64
#
```

Thanks!

Arnaldo Carvalho de Melo

acme@kernel.org

acme@redhat.com

linux-perf-users@vger.kernel.org