Overview of Recent Developments

Arnaldo Carvalho de Melo

Red Hat Inc.

San Diego
August, 2012
1 Recent Developments
   - Regression Tests
   - Annotation
   - DWARF CFI callchains
   - Report improvements
   - GTK UI
   - Embedded Platforms

2 Scripting
   - Available Scripts
   - Generate Scripts
Regression Tests

1. ‘perf test’
2. Growing number of tests
3. Needs to be hooked to git request-pull
4. New features should come with associated tests
5. Jiri Olsa has been doing that!
Recent Developments
Scripting
Regression Tests
Annotation
DWARF CFI callchains
Report improvements
GTK UI
Embedded Platforms

Current Tests

[root@sandy ~]# perf test
  1: vmlinux symtab matches kallsyms: Ok
  2: detect open syscall event: Ok
  3: detect open syscall event on all cpus: Ok
  4: read samples using the mmap interface: Ok
  5: parse events tests: Ok
  6: x86 rdpmc test: Ok
  7: Validate PERF_RECORD_* events & perf_sample fields: Ok
  8: Test perf pmu format parsing: Ok
  9: Test dso data interface: Ok
[root@sandy ~]#
Annotation

1. Instruction augmentation
2. Lines connecting jumps
3. Toggling features
4. Static or Live modes
5. Needs to support multiple events on same screen
DWARF CFI callchains

1. For -fomit-frame-pointer binaries
2. Copies chunks of userspace stack/regs
3. CFI post processed
4. Example later in the presentation
5. Contributed by Jiri Olsa
Report improvements

1. Add sort by source file:line number, using addr2line
2. Should use MiniDebuginfo
3. Print snapshots to file for easier mailing around
4. Expand just the callchains of interest, snapshot
Recent Developments

Regression Tests
Annotation
DWARF CFI callchains
Report improvements
GTK UI
Embedded Platforms

Report srcline example

```
[root@sandy ~]# perf record find / > /dev/null
^C[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.225 MB perf.data (~9815 samples)

[root@sandy ~]# perf report -s srcline,sym | grep -v ^# | head -5
7.52% security/selinux/ss/avtab.c:185 [k] avtab_search_node
1.48% kernel/spinlock.c:385 [k] _raw_spin_lock
1.24% security/selinux/ss/avtab.c:173 [k] avtab_search_node
1.09% security/selinux/ss/context.h:152 [k] sidtab_context_to_sid
0.62% kernel/spinlock.c:385 [k] _raw_spin_lock_irqsave
[root@sandy ~]#
```
Contributed by Pekka Emberg

Being improved by Namhyung Kim

TUI code being refactored to be used in GUI
Embedded Platforms

1. Patches to fix cross compilation
2. Android support
3. Add missing functions
4. Both in tools/perf and in upstream Bionic
UI - TODO

1. Allow selecting events to record at any time
2. Start with top
3. Freeze == report
4. Save == record
5. Integrate with perf probe
6. Go from annotate to probe, restart top
Scripting

1. Use scripting languages to process events
2. Python and Perl
3. Allows tapping into tons of language libraries
4. Several scripts available
5. Generate scripts from perf.data
6. General events support added by Feng Tang/Robert Richter
<table>
<thead>
<tr>
<th>Script Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>syscall-counts-by-pid [comm]</td>
<td>system-wide syscall counts, by pid</td>
</tr>
<tr>
<td>sctop [comm] [interval]</td>
<td>syscall top</td>
</tr>
<tr>
<td>failed-syscalls-by-pid [comm]</td>
<td>system-wide failed syscalls, by pid</td>
</tr>
<tr>
<td>net_dropmonitor</td>
<td>display a table of dropped frames</td>
</tr>
<tr>
<td>sched-migration</td>
<td>sched migration overview</td>
</tr>
<tr>
<td>netdev-times [tx] [rx] [dev=] [debug]</td>
<td>display a process of packet and process</td>
</tr>
<tr>
<td>futex-contention</td>
<td>futext contention measurement</td>
</tr>
<tr>
<td>syscall-counts [comm]</td>
<td>system-wide syscall counts</td>
</tr>
<tr>
<td>rw-by-pid</td>
<td>system-wide r/w activity</td>
</tr>
<tr>
<td>rwtop [interval]</td>
<td>system-wide r/w top</td>
</tr>
<tr>
<td>workqueue-stats</td>
<td>workqueue stats (ins/exe/create/destroy)</td>
</tr>
<tr>
<td>rw-by-file &lt;comm&gt;</td>
<td>r/w activity for a program, by file</td>
</tr>
<tr>
<td>failed-syscalls [comm]</td>
<td>system-wide failed syscalls</td>
</tr>
<tr>
<td>wakeup-latency</td>
<td>system-wide min/max/avg wakeup latency</td>
</tr>
</tbody>
</table>

 Arnaldo Carvalho de Melo  
 perf tools
Generate Scripts

1. From the events found in perf.data file
2. Quickly start writing event handling
3. Creates function skeletons for each trace event
4. With a common set of parameters
5. Plus event specific parameters
6. Calls methods at init, exit and for unhandled events
7. Comes with library of tracing specific methods
Listing Possible probe points

```
[root@ana icmp]# perf probe -L icmp_rcv
<icmp_rcv:0>
  0  int icmp_rcv(struct sk_buff *skb)
   1  {

59   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.
      */
      66   if ((icmph->type == ICMP_ECHO ||
               icmph->type == ICMP_TIMESTAMP) &&
              net->ipv4.sysctl_icmp_echo_ignore_broadcasts)
           goto error;
     }

71   if (icmph->type != ICMP_ECHO &&
```
Listing variables that can be collected

[root@ana ~]# perf probe -V icmp_rcv:66
Available variables at icmp_rcv:66
  @<icmp_rcv+343>
    struct icmphdr* icmph
    struct net*     net
    struct rtable*  rt
    struct sk_buff* skb

[root@ana ~]#
Adding a probe

[root@ana icmp]# perf probe icmp_rcv:66 'type=icmph->type'
Add new event:
    probe:icmp_rcv    (on icmp_rcv:66 with type=icmph->type)

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

    perf record -e probe:icmp_rcv -aR sleep 1

[root@ana ~]# perf probe --list
    probe:icmp_rcv (on icmp_rcv:66@net/ipv4/icmp.c with type)

[root@ana icmp]# perf record -a -g -e probe:icmp_rcv
^C[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.324 MB perf.data ]
Generating a python script from perf.data

```
[root@ana icmp]# perf script -g python
generated Python script: perf-trace.py

[root@ana icmp]# cat perf-trace.py

def trace_begin():
  print "in trace_begin"

def trace_end():
  print "in trace_end"

def probe__icmp_rcv(evname, cpu, secs, nsecs, pid, comm, probe_ip, type):
  print "%s %u.%u type=%u" % (evname, secs, nsecs, type)
```
Recent Developments
Scripting

Available Scripts
Generate Scripts

Running python script

[root@ana icmp]# perf script -s perf-trace.py
in trace_begin
probe__icmp_rcv 71171.964568380 type=8
probe__icmp_rcv 71177.792382154 type=8
probe__icmp_rcv 71178.792236953 type=8
in trace_end
[root@ana icmp]#
Backtraces from probes

[root@ana ~]# perf report --stdio
# Events: 2
#
# Overhead  Command  Shared Object  Symbol
#
100.00%  ping  [kernel.kallsyms]  [k] icmp_rcv
|
--- icmp_rcv
   ip_local_deliver_finish
   NF_HOOK.clone.1
   ip_local_deliver
   ip_rcv_finish
   NF_HOOK.clone.1
   ip_rcv
   __netif_receive_skb
   process_backlog
   net_rx_action
   __do_softirq
   0xb7707424

[root@ana ~]#
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
[root@sandy ~]#
```
Adding userspace probe

[root@sandy ~]# 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

[root@sandy ~]#
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 samples) ]
#
[root@sandy ~]# 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
[root@sandy ~]#
Recent Developments

Scripting

Available Scripts

Generate Scripts

Verbose report snapshot

[root@sandy ~]# 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
[root@sandy ~]# rpm -qf ‘which sleep’
coreutils-8.4-19.el6.x86_64
[root@sandy ~]# rpm -q coreutils-debuginfo
package coreutils-debuginfo is not installed
[root@sandy ~]# rpm -q glibc-debuginfo
glibc-debuginfo-2.12-1.80.el6_3.4.x86_64
[root@sandy ~]#
ETOOMANYLIBS

1. GTK UI needs to be separate binary
2. Use Kconfig to select desired features
3. Minimal tool just for recording/top
4. RFC patch from David Ahern
5. Make 'perf script' use dlopen according to script lang
That is all folks!

Thanks!

Arnaldo Carvalho de Melo

acme@infradead.org

acme@redhat.com

linux-perf-users@vger.kernel.org