Perf Tools: Recent Improvements
Recent developments and discussion about TODO

Arnaldo Carvalho de Melo

Red Hat Inc.

Netconf and Linux Plumbers Conference, Cambridge
November, 2010
1. **Improvements on initial set of tools**
   - Tool Integration
   - Slang Based TUI

2. **New Tools**
   - perf diff
   - perf archive
   - perf probe

3. **Scripting**
   - Available Scripts
   - Generate Scripts

4. **KVM Support**

5. **Work in Progress**

6. **That is all folks!**
Improvements on initial set of tools

1. Tools Integration
2. Slang based Text User Interface
3. Use of build ids
Tools Integration

1. One tool doesn’t do it all
2. Combine steps to achieve multiple results
3. Allows spreading work flows over multiple machines
4. Profiling fast path
5. Report to annotate
6. Reuse perf.data parsing
Slang Based TUI

1. GUIs not necessarily better
2. We still have mutt and pine users, after all
3. But the changes paves the way for GUIs
4. mutt like interface
5. report to annotate fast path
6. Zoom in/out DSOs/threads
7. Keys used: arrows + ENTER mostly, TAB sometimes
8. Still don’t like it? Use –stdio
Improvements on initial set of tools

- New Tools
- Scripting
- KVM Support

Work in Progress
That is all folks!

Tool Integration
Slang Based TUI

**perf report TUI**

```
Events: 4K cycles
+  6.73%  wget [kernel.kallsyms] [k] copy_to_user
+  5.64%  wget [kernel.kallsyms] [k] __copy_from_user_11_nozero
-  2.91%  wget [kernel.kallsyms] [k] __might_sleep
  - __might_sleep
    + 16.40%  might_fault
    + 13.36%  __getblk
    + 12.63%  lock_buffer
    + 10.26%  slab_pre_alloc_hook
    +  6.41%  ext4_mark_inode_dirty
    +  5.52%  lock_page
    +  5.48%  set_fd_set
    +  5.35%  do_select
    +  4.39%  down_read
    +  3.79%  generic_file_buffered
    +  3.18%  mutex_lock
    +  1.92%  do_get_write_access
    +  1.72%  generic_file_aio_write
    +  1.54%  copy_to_user
    +  1.42%  n_tty_write
    +  1.15%  core_sys_select
    +  1.13%  ext4_dirty_inode
    +  1.03%  __ext4_get_inode_loc
    +  0.90%  kmem_cache_alloc
    +  0.87%  __alloc_pages_nodemask
    +  0.79%  unmmap_underlying_metadata

For a higher level overview, try: perf report --sort comm,dso
```
Improvements on initial set of tools
New Tools
Scripting
KVM Support
Work in Progress
That is all folks!

Tool Integration
Slang Based TUI

perf annotate

1. Starts at the line with most hits
2. Tabs through ordered list of hot lines
Improvements on initial set of tools

New Tools
- Scripting
- KVM Support

Work in Progress
- That is all folks!

Tool Integration
- Slang Based TUI

perf annotate TUI

```
might_sleep

return (struct thread_info *)
    (current_stack_pointer & ~(THREAD_SIZE - 1));

0.80 :   
0.00 :   
0.00 :   
}

#ifdef CONFIG_DEBUG_SPINLOCK_SLEEP
static inline int preempt_count_equals(int preempt_offset)
{
    int nested = (preempt_count() & ~PREEMPT_ACTIVE) + rcu_preempt_depth();

1.60 :   
26.40 :   

void __might_sleep(const char *file, int line, int preempt_offset)
{
    #ifdef in_atomic
        static unsigned long prev_jiffies;  /* ratelimiting */

4.00 :   
6.40 :   

    if ((preempt_count_equals(preempt_offset) && !irqs_disabled()) ||

    c042b537: 39 c8
    c042b575: 75 0f

#define __PV_IS_CALLEE_SAVE(func)
    ((struct paravirt_caller_save) { func })

    static inline unsigned long arch_local_save_flags(void)

<, -> or ESC: exit, TAB/shift+TAB: cycle thru samples
```
Improvements on initial set of tools
New Tools
Scripting
KVM Support
Work in Progress
That is all folks!

Tool Integration
Slang Based TUI

UI - TODO

1. perf top
2. Allow selecting events to record at any time
3. Start with top
4. Freeze == report
5. Save == record
6. perf probe
7. Go from annotate to probe, restart top
Improvements on initial set of tools

New Tools
Scripting
KVM Support

Work in Progress
That is all folks!

Tool Integration
Slang Based TUI

Perf top

Considers user space symbols too:

PerfTop: 155 irqs/sec kernel:83.9% [1000Hz cycles], (all, 2 CPUs)

<table>
<thead>
<tr>
<th>samples</th>
<th>pcnt</th>
<th>function</th>
<th>DSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>119.00</td>
<td>12.0%</td>
<td>read_hpet</td>
<td>[kernel]</td>
</tr>
<tr>
<td>43.00</td>
<td>4.4%</td>
<td>__strstr_ia32</td>
<td>/lib/libc-2.12.1.so</td>
</tr>
<tr>
<td>28.00</td>
<td>2.8%</td>
<td>system_call</td>
<td>[kernel]</td>
</tr>
<tr>
<td>25.00</td>
<td>2.5%</td>
<td>unix_poll</td>
<td>[kernel]</td>
</tr>
<tr>
<td>24.00</td>
<td>2.4%</td>
<td>aes_enc_blk</td>
<td>[aes_i586]</td>
</tr>
<tr>
<td>21.00</td>
<td>2.1%</td>
<td>schedule</td>
<td>[kernel]</td>
</tr>
<tr>
<td>21.00</td>
<td>2.1%</td>
<td>_raw-spin_lock_IRQsave</td>
<td>[kernel]</td>
</tr>
<tr>
<td>19.00</td>
<td>1.9%</td>
<td>_raw-spin_unlock_IRQrestore</td>
<td>[kernel]</td>
</tr>
<tr>
<td>19.00</td>
<td>1.9%</td>
<td>aes_dec_blk</td>
<td>[aes_i586]</td>
</tr>
<tr>
<td>18.00</td>
<td>1.8%</td>
<td>probe_workqueue_insertion</td>
<td>[kernel]</td>
</tr>
<tr>
<td>17.00</td>
<td>1.7%</td>
<td>hpet_next_event</td>
<td>[kernel]</td>
</tr>
<tr>
<td>13.00</td>
<td>1.3%</td>
<td>fget_light</td>
<td>[kernel]</td>
</tr>
<tr>
<td>13.00</td>
<td>1.3%</td>
<td>do_select</td>
<td>[kernel]</td>
</tr>
<tr>
<td>12.00</td>
<td>1.2%</td>
<td>audit_syscall_entry</td>
<td>[kernel]</td>
</tr>
<tr>
<td>12.00</td>
<td>1.2%</td>
<td>ktime_get</td>
<td>[kernel]</td>
</tr>
<tr>
<td>11.00</td>
<td>1.1%</td>
<td>test_tii_thread_flag</td>
<td>[kernel]</td>
</tr>
<tr>
<td>11.00</td>
<td>1.1%</td>
<td>std::List_node_base::transfer(std::L</td>
<td>libstdc++.so.6.0.13</td>
</tr>
<tr>
<td>11.00</td>
<td>1.1%</td>
<td>native_sched_clock</td>
<td>[kernel]</td>
</tr>
<tr>
<td>11.00</td>
<td>1.1%</td>
<td>vsnprintf</td>
<td>[kernel]</td>
</tr>
<tr>
<td>11.00</td>
<td>1.1%</td>
<td>format_decode</td>
<td>[kernel]</td>
</tr>
<tr>
<td>10.00</td>
<td>1.0%</td>
<td>index</td>
<td>/lib/libc-2.12.1.so</td>
</tr>
</tbody>
</table>

Arnaldo Carvalho de Melo
Perf Tools: Recent Improvements
perf stat

1. List of CPUs to monitor
2. Ask for precise events (PEBS) using suffix: ",-e cycles:p"
3. Multiple 'p' characters == more precise
4. Proof of concept patch for printing counters periodically ready
5. Merge app log output sorting by timestamps
New Tools

Introduced after Plumbers’2009:

1. diff
2. archive
3. probe
4. trace
5. several trace ones (timechart, etc)
perf diff

1. Shows difference in symbol hits between two perf.data files
2. Keyed by build-ids in the cache
3. Should support more than two files
4. Generating version X samples symbol plottings
5. Read ”Differential Profiling” paper by Paul McKenney on how to use it
Perf Tools: Recent Improvements

- Improvements on initial set of tools
- New Tools
  - Scripting
  - KVM Support
  - Work in Progress
  - That is all folks!

**perf archive**

1. Looks at perf.data files for DSOs with hits
2. Creates tarball
3. Transfer to another machine
4. Populate the cache
5. Use report and annotate
6. Handles endianness

Arnaldo Carvalho de Melo
**Perf probe**

1. Inserts dynamic probes
2. Doesn’t necessarily require debuginfo
3. Can collect variables
4. Struct members can be specified to any level
5. Works with callchains
6. Works on the core kernel and on modules
7. Supports wildcards in probe names
8. Together with perf trace == systemtap subset
9. Example of use together with scripting later in this presentation
10. Contributed by Masami Hiramatsu
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. Contributed by Tom Zanussi
Available Scripts

[root@ana ~]# perf trace --list
List of available trace scripts:

- **rw-by-pid**: system-wide r/w activity
- **wakeup-latency**: system-wide min/max/avg wakeup latency
- **workqueue-stats**: workqueue stats (ins/exe/create/destroy)
- **rwtop [interval]**: system-wide r/w top
- **failed-syscalls [comm]**: system-wide failed syscalls
- **rw-by-file <comm>**: r/w activity for a program, by file
- **syscall-counts-by-pid [comm]**: system-wide syscall counts, by pid
- **netdev-times [tx] [rx] [dev=}**: display a process of packet and processing time
- **sctop [comm] [interval]**: syscall top
- **futex-contention**: futex contention measurement
- **sched-migration**: sched migration overview
- **failed-syscalls-by-pid [comm]**: system-wide failed syscalls, by pid
- **syscall-counts [comm]**: system-wide syscall counts

[Arnaldo Carvalho de Melo]

Perf Tools: Recent Improvements
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
Improvements on initial set of tools
New Tools
Scripting
KVM Support
Work in Progress
That is all folks!

Available Scripts
Generate Scripts

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 ((icmphy->type == ICMP_ECHO ||
      icmphy->type == ICMP_TIMESTAMP) &&
      net->ipv4.sysctl_icmp_echo_ignore_broadcasts) {
      goto error;
   }
71 if (icmphy->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=icmphp->type'
Add new event:
probe:icmp_rcv (on icmp_rcv:66 with type=icmphp->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 trace -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)
Running python script

```
[root@ana icmp]# perf trace -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]#
```
Improvements on initial set of tools
New Tools
Scripting
KVM Support
Work in Progress
That is all folks!

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 ~]#
```
Scripting TODO List

1. Convert trace builtins to scripts (sched, kmem, etc)
2. Convert net/ipv4/tcp_probe.c
3. SCTP and DCCP variants too
4. Write more scripts for showing where IO is happening
5. Improve passing data from record to trace
6. Remove requirement on using netcat for dual machine use
7. Write more scripts (you can help here!)
KVM Support

1. Collect guest OS statistics from host side.
2. `top`, `record`, `report`, `diff`, `buildid-list`
3. Need to specify guest `vmlinux` or `kallsyms`, `/proc/modules`
4. Or `–guestmount` directory with `sshfs` mounted per pid subdirs
5. Use `–pid` to specify specific guest
6. Contributed by Zhang, Yanmin.
Improvements on initial set of tools
New Tools
Scripting
KVM Support
Work in Progress
That is all folks!

perf top kvm example

```
# perf kvm --host --guest --guestkallsyms=guest/kallsyms 
  --guestmodules=guest/modules top

PerfTop: 16010 irqs/sec kernel:59.1% us: 1.5% guest 
    kernel:31.9% guest us:7.5% [+1000Hz cycles]

samples pcnt function DSO
--------- ----- --------------------------- ---------
38770.00 20.4% __ticket_spin_lock [guest.kernel]
22560.00 11.9% ftrace_likely_update [kernel]
  9208.00  4.8% __lock_acquire [kernel]
  5473.00  2.9% trace_hardirqs_off_caller [kernel]
  5222.00  2.7% copy_user_generic_string [guest.kernel]
  4450.00  2.3% validate_chain [kernel]
  4262.00  2.2% trace_hardirqs_on_caller [kernel]
  4239.00  2.2% do_raw_spin_lock [kernel]
  3548.00  1.9% do_raw_spin_unlock [kernel]
  2487.00  1.3% lock_release [kernel]
  2165.00  1.1% __local_bh_disable [kernel]
  1905.00  1.0% check_chain_key [kernel]
```

Arnaldo Carvalho de Melo

Perf Tools: Recent Improvements
Improvements on initial set of tools

New Tools

Scripting

KVM Support

Work in Progress

That is all folks!

Work in Progress

1. cgroups support
2. utrace to probe user space
3. PerfKit GUI
4. In addition to KernelShark and sysprof GUls
Thanks!

Arnaldo Carvalho de Melo

acme@infradead.org

acme@redhat.com

linux-perf-users@vger.kernel.org