Linux perf tools

Recent changes and improvements

Namhyung Kim

LG Electronics

Korea Linux Forum 2012

1. Introduction

2. Report Improvements
   - Annotation
   - Diff

3. New Features
   - Uprobes
   - Trace

4. Advanced Usage
   - Scripting
   - KVM Support

5. Internal Changes
   - DWARF CFI callchains
   - Embedded Systems

Introduction to Perf

- Modern, actively-developed performance analysis tool
- Monitoring H/W PMU counter as well as S/W events
- Counting or sampling
- GIT-like architecture
- NO daemons required
- Callchains (stack backtrace) supported
- perf.data file per session
- Support for perl/python scripting

Specifying events

- 'perf list' to get a full list of supported events
- Pre-defined (architectural) H/W events
- S/W and tracepoint events
- H/W Raw events (hex numbers)
- H/W breakpoint events
- PMU events (symbols)
- Event groups

```
cycles
r003c
cpu/event=0x3c,umask=0/

uncore_imc_0/event=0x4,umask=0x3/
uncore_imc_0/cas_count_read/

{branches,branch-misses}
```

Specifying targets

- Processes (-p *pids*)
- Threads (-t *tids*)
- User (-u *uid*)
- System-wide (-a)
- Cpus (-C *cpus*)
- Cgroups (-G *cgroup*)
- Workload

Basic usage

- Counting number of events
  - perf stat [*events*] *target*
- Collecting event samples
  - perf record [*events*] *target*
- Analyzing the samples
  - perf report
- Live Monitoring
  - perf top [*events*] [*target*]

Report improvements

- Sort by selected keys: pid, comm, dso, symbol and parent
  - Similar to 'group-by' keyword in SQL
  - New sort key - source file:line number, using addr2line
- Write snapshot to a file on TUI
  - For easy mailing around
  - Expand just the callchains of interest
- GTK2 based GUI supported
- Event group view patch submitted

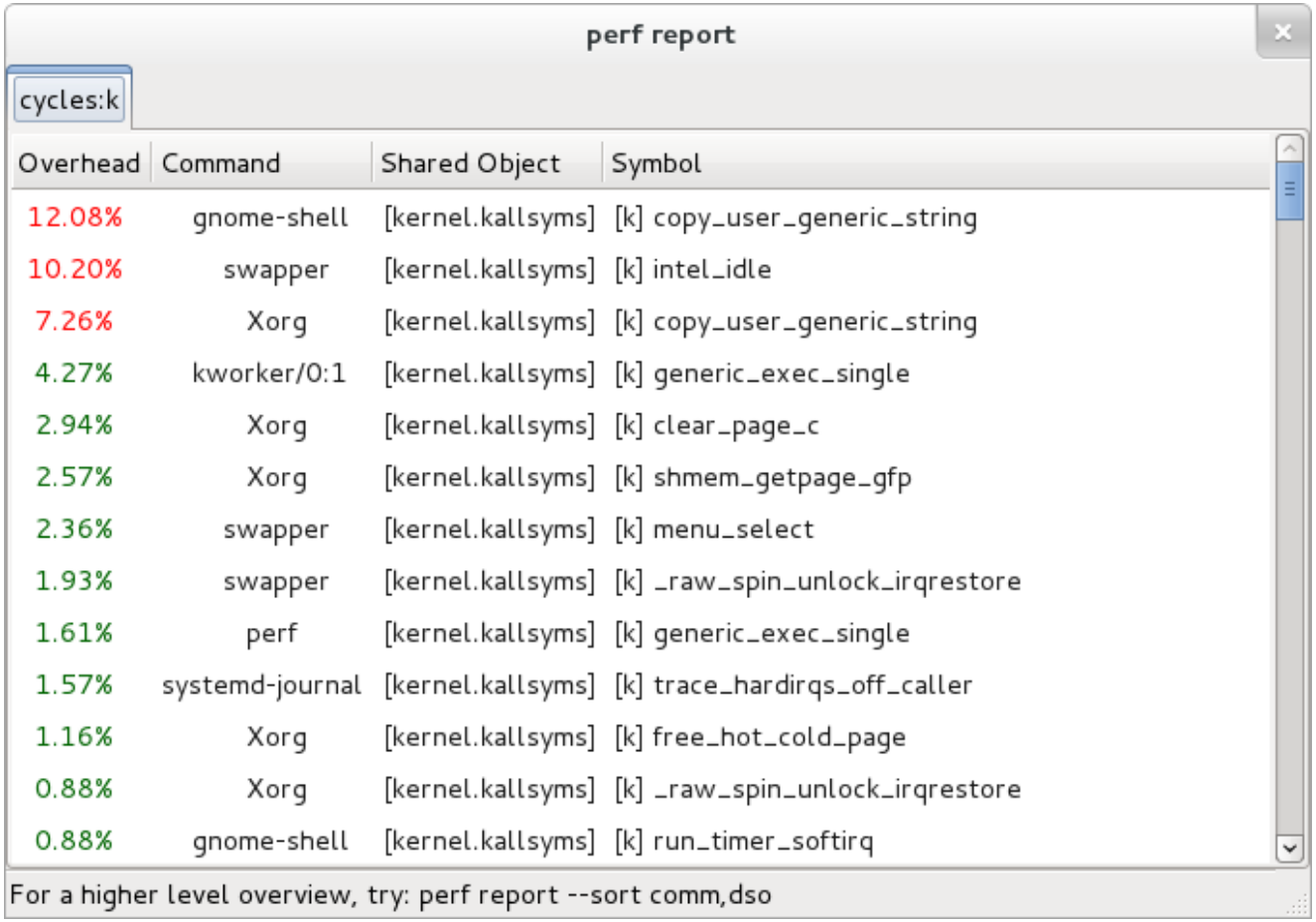
Report srcline example

```
# perf record -a -e cycles:k sleep 1
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.277 MB perf.data (~12101 samples) ]
#
# perf report -s srcline,sym | grep -v ^# | head
    19.25%  arch/x86/lib/copy_user_64.S:243       [k] copy_user_generic_string
     9.65%  drivers/idle/intel_idle.c:311         [k] intel_idle
     4.70%  kernel/smp.c:104                      [k] generic_exec_single
     2.94%  arch/x86/lib/clear_page_64.S:13       [k] clear_page_c
     2.45%  mm/shmem.c:1235                       [k] shmem_getpage_gfp
     1.93%  arch/x86/include/asm/paravirt.h:832   [k] _raw_spin_unlock_irqrestore
     1.69%  drivers/cpuidle/governors/menu.c:152  [k] menu_select
     1.57%  kernel/trace/trace_irqsoff.c:510      [k] trace_hardirqs_off_caller
     1.22%  arch/x86/include/asm/spinlock.h:54    [k] _raw_spin_lock
     0.74%  arch/x86/include/asm/processor.h:668  [k] generic_exec_single
```

Report snapshot example

```
# 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
```

Report GTK2 UI screenshot



Annotation

- Analyze overhead at instruction level
  - Use objdump for disassembly
- Annotate using a compact format
  - Instruction augmentation
  - Hide uninteresting information
  - Arrows/Lines connecting jumps
  - TUI only

Compact annotation

Change default objdump output

```
       |400662:  callq  4004a0 <alarm@plt>
       |400667:  jmp    400675 <main+0x66>
       |400669:  mov    -0x10(%rbp),%rax
 66.68 |40066d:  add    $0x1,%rax
 15.83 |400671:  mov    %rax,-0x10(%rbp)
 17.49 |400675:  mov    0x20041d(%rip),%eax        # 600a98 <__TMC_END__>
```

into

```
       |      callq  alarm@plt
       |      jmp    66
       |5a:   mov    -0x10(%rbp),%rax
 66.68 |      add    $0x1,%rax
 15.83 |      mov    %rax,-0x10(%rbp)
 17.49 |66:   mov    __TMC_END__,%eax
```

Differential Profiling

- 'perf diff' for comparing performance results
  - perf record
  - *do your job...*
  - perf record
  - perf diff
- Differential profiling
  - paper from Paul McKenney
  - Delta profiling (default)
  - Ratio differential profiling
  - Weighted differential profilng

New features

- Uprobes
  - User-space counterpart of kprobes
  - Breakpoint for user apps, handled in kernel
  - Requires root privilege
  - Merged on v3.5
- 'perf trace'
  - Easy to use and straightforward tracing tool
  - Using perf's infrastructure and workflow
  - Will be available from v3.7

Uprobes

- Listing probeable functions in userspace DSO

```
# perf probe -F /lib64/libc-2.15.so | grep ^m | head -5
madvise
malloc
malloc@plt
malloc_info
mblen
```

- Adding userspace probe

```
# perf probe -x /lib64/libc-2.15.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
```

Trace

- New 'perf trace' command for tracing system call events
- Aim at strace-like easy to use tool
- Support live mode only currently
- Will include more information like page faults
- Initial work by Thomas Gleixner and Ingo Molnar
  - `http://lwn.net/Articles/415728/`

Proposed page fault tracing

```
# perf trace --pagefaults usleep 1
     0.000:  ... [continued]: read()) = 0
     0.025:    #PF  1: [            3099c7f040]: => /lib/modules/3.6.0-rc5+/build/vmlinux offset: 0x3118c7f040
     0.038:    #PF  2: [            3099c35250]: => 0000003099c35250 (R.U)
     0.047:    #PF  3: [            3099c80b40]: => 0000003099c80b40 (R.U)
     0.055:    #PF  4: [            3099c80b51]: => 0000003099d54821 (R.U)
     0.064:    #PF  5: [            3099c82590]: => 0000003099c82590 (R.U)
...
     1.288:    #PF  8: [            3099cac727]: => /lib64/ld-2.15.so offset: 0x220fa8 page: 544 (W.K)
     1.314:    #PF  9: [            3099cac727]: => 00007ffffebe5ef9 (W.K)
     1.343: execve(arg0: 140734116673493, arg1: 140734116683888, arg2: 32104608, arg3: 140734116673024, arg4: 7
...
     1.775: open(filename: 140737467273824, flags: 0, mode: 0                    ) = -2
     1.785: stat(filename: 140737467273824, statbuf: 140737467274000             ) = 0
     1.794:    #PF 41: [  _dl_sysdep_read_whole_file]: => /lib64/ld-2.15.so offset: 0xf670 page: 15 (R.U)
     1.810: open(filename: 208733843841, flags: 0, mode: 1                       ) = 3
     1.818: fstat(fd: 3, statbuf: 140737467273920                                ) = 0
     1.835: mmap(addr: 0, len: 91882, prot: 1, flags: 2, fd: 3, off: 0           ) = -998703104
     1.843: close(fd: 3                                                          ) = 0
     1.852:    #PF 42: [ _dl_load_cache_lookup]: => 00007f4ac4790000 (R.U)
     1.860:    #PF 43: [                memcmp]: => 00007f4ac4793508 (R.U)
     1.868:    #PF 44: [ _dl_load_cache_lookup]: => 00007f4ac4796a4c (R.U)
     1.876:    #PF 45: [      _dl_cache_libcmp]: => 00007f4ac47a00a4 (R.U)
     1.884:    #PF 46: [ _dl_load_cache_lookup]: => 00007f4ac4794fac (R.U)
     1.892:    #PF 47: [      _dl_cache_libcmp]: => 00007f4ac479d098 (R.U)
     1.899:    #PF 48: [      _dl_cache_libcmp]: => 00007f4ac479b770 (R.U)
     1.907:    #PF 49: [      _dl_cache_libcmp]: => 00007f4ac479c3ef (R.U)
...
```

Scripting

- Use script languages to process events
- Currently perl and python are supported
  - Require development packages installed
- Several scripts are available
- Run existing script to perf.data
- Generate script from perf.data
- Support non-tracepoint event samples
- Script browser patch submitted

Available scripts

```
# perf script --list
List of available trace scripts:
  netdev-times [tx] [rx] [dev=] [debug] display a process of packet and processing time
  failed-syscalls-by-pid [comm]        system-wide failed syscalls, by pid
  net_dropmonitor                      display a table of dropped frames
  futex-contention                     futext contention measurement
  syscall-counts [comm]                system-wide syscall counts
  syscall-counts-by-pid [comm]         system-wide syscall counts, by pid
  event_analyzing_sample               analyze all perf samples
  sctop [comm] [interval]              syscall top
  sched-migration                      sched migration overview
  wakeup-latency                       system-wide min/max/avg wakeup latency
  rw-by-pid                            system-wide r/w activity
  rw-by-file <comm>                    r/w activity for a program, by file
  failed-syscalls [comm]               system-wide failed syscalls
  rwtop [interval]                     system-wide r/w top
  workqueue-stats                      workqueue stats (ins/exe/create/destroy)
```

Generating script

```
# perf record -a -e sched:sched_wakeup sleep 1
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.351 MB perf.data (~15347 samples) ]
#
# perf script -g python
generated Python script: perf-script.py
#
# cat perf-script.py
def trace_begin():
        print "in trace_begin"

def trace_end():
        print "in trace_end"

def sched__sched_wakeup(event_name, context, common_cpu,
        common_secs, common_nsecs, common_pid, common_comm,
        comm, pid, prio, success,
        target_cpu):
                print_header(event_name, common_cpu, common_secs, common_nsecs,
                        common_pid, common_comm)

                print "comm=%s, pid=%u, prio=%u, " \
                "success=%u, target_cpu=%u\n" % \
                (comm, pid, prio, success,
                target_cpu),

def print_header(event_name, cpu, secs, nsecs, pid, comm):
        print "%-20s %5u %05u.%09u %8u %-20s " % \
        (event_name, cpu, secs, nsecs, pid, comm),
```

Running script

```
# perf script -s perf-script.py
in trace_begin
sched__sched_wakeup    3 24794.823739899  13044 perf         comm=perf, pid=13045, prio=120, success=1, target_
sched__sched_wakeup    9 24794.823800642  13045 perf         comm=migration/9, pid=50, prio=0, success=1, targe
sched__sched_wakeup    0 24794.824777438   1412 gnome-shell  comm=kworker/0:1, pid=75, prio=120, success=1, tar
sched__sched_wakeup    5 24794.826330919   1413 gnome-shell  comm=gnome-shell, pid=1411, prio=120, success=1, t
sched__sched_wakeup    5 24794.826332752   1413 gnome-shell  comm=gnome-shell, pid=1414, prio=120, success=1, t
sched__sched_wakeup    5 24794.826333622   1413 gnome-shell  comm=gnome-shell, pid=1409, prio=120, success=1, t
sched__sched_wakeup    5 24794.826334675   1413 gnome-shell  comm=gnome-shell, pid=1416, prio=120, success=1, t
sched__sched_wakeup    5 24794.826335456   1413 gnome-shell  comm=gnome-shell, pid=1415, prio=120, success=1, t
sched__sched_wakeup    5 24794.826336199   1413 gnome-shell  comm=gnome-shell, pid=1412, prio=120, success=1, t
sched__sched_wakeup    5 24794.826336899   1413 gnome-shell  comm=gnome-shell, pid=1410, prio=120, success=1, t
sched__sched_wakeup   10 24794.826376847   1409 gnome-shell  comm=gnome-shell, pid=1397, prio=120, success=1, t
sched__sched_wakeup    1 24794.826475446   1397 gnome-shell  comm=Xorg, pid=940, prio=120, success=1, target_cp
sched__sched_wakeup    2 24794.826895617    940 Xorg         comm=gnome-shell, pid=1397, prio=120, success=1, t
sched__sched_wakeup    3 24794.826929411   1397 gnome-shell  comm=Xorg, pid=940, prio=120, success=1, target_cp
sched__sched_wakeup    4 24794.826970603    940 Xorg         comm=gnome-shell, pid=1397, prio=120, success=1, t
sched__sched_wakeup    4 24794.826973342    940 Xorg         comm=firefox, pid=10371, prio=120, success=1, targ
sched__sched_wakeup    0 24794.827043688  10371 firefox      comm=firefox, pid=10342, prio=120, success=1, targ
sched__sched_wakeup    4 24794.827087472    940 Xorg         comm=gnome-shell, pid=1397, prio=120, success=1, t
sched__sched_wakeup    9 24794.827138617  10342 firefox      comm=firefox, pid=10371, prio=120, success=1, targ
...
in trace_end
```

It'd be wonderful if you submit your great script to upstream!

KVM Support

- Collect guest OS statistics from host
- Use –pid to specify specific guest
- Need to specify guest vmlinux, kallsyms or modules info
- Or –guestmount directory with sshfs mounted per pid subdirs
- 'perf kvm (top/record/report/diff/buildid-list)'
  - New 'stat' subcommand added recently
  - Currently vmexit, mmio, ioport events are supported

perf kvm stat

```
# pgrep qemu
12027
#
# perf kvm stat record -p 12027
^C[ perf record: Woken up 6 times to write data ]
[ perf record: Captured and wrote 12.837 MB perf.data.guest (~560846 samples) ]
#
# perf kvm stat report --event=vmexit


Analyze events for all VCPUs:

             VM-EXIT    Samples  Samples%     Time%         Avg time

      IO_INSTRUCTION      17094    30.12%     0.20%      7.24us ( +-   5.05% )
         APIC_ACCESS      13546    23.87%     0.10%      4.47us ( +-   4.58% )
       EPT_VIOLATION      10146    17.88%     0.02%      1.29us ( +-   4.54% )
   PAUSE_INSTRUCTION       6660    11.73%     0.01%      0.97us ( +-   1.89% )
                 HLT       3380     5.96%    99.63%  18632.56us ( +-   7.74% )
  EXTERNAL_INTERRUPT       2006     3.53%     0.03%      8.92us ( +-   8.10% )
           CR_ACCESS       1482     2.61%     0.01%      2.36us ( +-   1.83% )
   PENDING_INTERRUPT        827     1.46%     0.00%      1.37us ( +-   3.31% )
       EXCEPTION_NMI        773     1.36%     0.00%      2.54us ( +-   2.54% )
       EPT_MISCONFIG        587     1.03%     0.01%     10.32us ( +-   3.47% )
                CPUID        253     0.45%     0.00%      1.14us ( +-   4.04% )

Total Samples:56754, Total events handled time:63212778.18us.
```

Internal changes

- 'perf test' for regression tests
- Integrate libtraceevent
- Refactor UI related code
- Better support for separated debug symbols
- Improve tracepoint handling

Regression tests

```
# 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
10: roundtrip evsel->name check: Ok
11: Check parsing of sched tracepoints fields: Ok
12: Generate and check syscalls:sys_enter_open event fields: Ok
```

DWARF CFI callchains

- For tracking optimized binaries
  - GCC adds -fomit-frame-pointer for those
  - Kernel has frame pointers but user apps don't
- Use it with '-g dwarf[,size]' option
- Copies chunks of userspace stack/regs
- CFI post processed using libunwind

Embedded Systems

- Add support for better off-site/cross-built analysis
- Fix various cross build problems and runtime issues
- Android support
  - Also add missing functions to upstream Bionic

Acknowledgement

Thanks to following people for providing valuable comments and feedbacks:

- Arnaldo Carvalho de Melo
- David Ahern
- Jiri Olsa
- Minchan Kim
- Stephane Eranian