Perf tools updates and beyond

Namhyung Kim <namhyung@kernel.org>

Google

東京 2025
LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

Data type profiling

- Associate PMU samples to data type using DWARF
- New code annotation along with type info
- New memory output fields : op, mem, cache, snoop, dtlb
  - useful sort keys: type, typeoff, typecln, symoff
- instruction tracking updates to handle pointer arithmetics
  - handle container_of()
- Still a lot of rooms to improve
  - language and architecture support

Data type profiling

```
$ perf annotate --code-with-type --stdio
,,,
    0.00 :    e04f:        jae     0xe08c <elf_dynamic_do_Rela>
    0.00 :    e051:        nopw    %cs:(%rax,%rax)
    0.00 :    e05c:        nopl    (%rax)
   30.32 :    e060:        movq    (%rdx), %rsi         # data-type: Elf64_Rela +0 (r_offset)
   24.55 :    e063:        movl    0x8(%rdx), %ecx              # data-type: Elf64_Rela +0x8 (r_info)
    0.00 :    e066:        addq    %r12, %rsi
    0.00 :    e069:        cmpq    $0x26, %rcx
    0.00 :    e06d:        je      0xe079 <elf_machine_rela_relative>
    0.00 :    e06f:        cmpq    $0x8, %rcx
    0.00 :    e073:        jne     0x15ef <elf_machine_rela_relative>
   33.90 :    e079:        movq    0x10(%rdx), %rcx             # data-type: Elf64_Rela +0x10 (r_addend)
    0.00 :    e07d:        addq    $0x18, %rdx
    0.00 :    e081:        addq    %r12, %rcx
    0.00 :    e084:        movq    %rcx, (%rsi)         # data-type: Elf64_Addr +0
    0.00 :    e087:        cmpq    %rbx, %rdx
    0.00 :    e08a:        jb      0xe060 <elf_dynamic_do_Rela>
    0.00 :    e08c:        movq    0x208(%r15), %r11            # data-type: struct link_map +0x208 (l_info)
```

Data type profiling

```
$ perf mem record -- \
    perf test -w datasym
```

with a change in 'datasym' program to generate accesses to 2nd cache line

```
$ perf mem report -F overhead,cache,dtlb,type,typecln -H
...
#                ----------- Cache ----------    D-TLB
#     Overhead       L1      L2 L1-buf   Other   L?-Hit  Data Type / Data Type Cacheline
# ..................................................  ................................
#
    90.17%    90.2%    0.0%    0.0%    0.0%    90.2%    buf
       68.51%    68.5%    0.0%    0.0%    0.0%    68.5%     buf: cache-line 0
       21.67%    21.7%    0.0%    0.0%    0.0%    21.7%     buf: cache-line 1
     6.95%     6.9%    0.0%    0.0%    0.0%     6.9%    sig_atomic_t
        6.95%     6.9%    0.0%    0.0%    0.0%     6.9%     sig_atomic_t: cache-line 0
     1.27%     0.0%    1.2%    0.0%    0.0%     1.3%    Elf64_Rela
        1.27%     0.0%    1.2%    0.0%    0.0%     1.3%     Elf64_Rela: cache-line 0
     0.84%     0.0%    0.0%    0.8%    0.0%     0.8%    unsigned char
        0.84%     0.0%    0.0%    0.8%    0.0%     0.8%     unsigned char: cache-line 0
...
```

Data type profiling

- Sometimes applications don't have full DWARF
  - only have line number tables
- An idea
  - get filename, line and column number for samples
  - look up source code at the location
  - parse the source code
  - consult a language-server to get type and fields
  - how to find/sync/verify source codes and binaries?

Latency profiling

- Aka wall-clock profiling
  - theoretically one sample at a moment
  - like in a single-CPU machine
- Divide sample weight by the parallelism
  - track scheduler context switches
- Identify less parallel parts easily
  - which would contribute to latency more

Latency profiling

```
$ perf record --latency -- \
    make -C tools/perf
```

```
$ perf report --latency -s comm --percent-limit=0.5 --stdio
...
#
#  Latency  Overhead  Command
# ........  ........  ...............
#
    54.29%    80.84%  cc1
    21.89%     5.56%  python3
    12.99%     2.75%  ld
     3.12%     1.34%  cc1plus
     2.91%     1.76%  as
     0.81%     0.18%  llvm-config
     0.75%     0.68%  clang
     0.63%     0.56%  sh
     0.61%     4.82%  shellcheck
```

Latency profiling

- Currently for single origin (process)
  - global parallelism tracking
- System wide mode? Multiple origins?
  - process-level latency profiling
  - users can give origins manually
- How to track context switches?
  - tracking sched-switch system-wide can be overwhelming
  - idea to inject them per-CPU
    - before and after idle
    - using sample frequency

Deferred unwinding

- Capture user callstack when it goes back to userspace (and handle page faults)
  - The kernel support from v6.19
  - only works with the frame-pointer for now
- ABI changes:
  - New perf event attributes: defer_callchain, defer_output
  - New perf callchain context: PERF_CONTEXT_USER_DEFERRED (and a cookie)
  - New perf record format: PERF_RECORD_CALLCHAIN_DEFERRED
- Perf report will delay processing samples until it finds deferred callchains using cookies

```
$ perf record --call-graph fp,defer -a sleep 1
```

Deferred unwinding

- What if task went to sleep before going to userspace?
- And profiling finishes while tasks are sleeping or in the kernel mode?
- Can we do this (in the kernel)?
    - save callchains when it goes to sleep
    - share the callchain (cookie) until it returns to userspace
- Maybe it can also defer collecting other sample data
    - like stack and registers (for DWARF unwinding)

Events and Metrics

- All descriptions are in JSON
  - vendor defined metrics
  - python support to write metrics
  - better reference for external usages
- Event parsing wildcard match
- More PMU information from the kernel?
  - core/uncore relation - hybrid core PMUs, multiple uncore PMUs
  - capability: sampling, modifiers?
  - fdinfo

Events and Metrics

```
$ perf stat -- perf test -w noploop 1

 Performance counter stats for 'perf test -w noploop 1':

                31      context-switches                 #     30.6 cs/sec  cs_per_second
                 0      cpu-migrations                   #      0.0 migrations/sec  migrations_per_second
             3,613      page-faults                      #   3572.1 faults/sec  page_faults_per_second
          1,011.45 msec task-clock                       #      1.0 CPUs  CPUs_utilized
           130,344      branch-misses                    #      0.0 %  branch_miss_rate         (88.77%)
     6,517,662,940      branches                         #   6443.9 M/sec  branch_frequency     (88.93%)
     4,378,161,108      cpu-cycles                       #      4.3 GHz  cycles_frequency       (88.94%)
    25,979,821,938      instructions                     #      5.9 instructions  insn_per_cycle  (88.93%)
                        TopdownL1                        #      0.5 %  tma_backend_bound
                                                         #     10.0 %  tma_bad_speculation      (88.93%)
                                                         #      9.7 %  tma_frontend_bound       (77.57%)
                                                         #     79.9 %  tma_retiring             (88.64%)
```

Lock contention profiling

- Slab object lock symbolization using BPF
  - still missing type info
  - offset in the type also needed
- Lock delay injection
  - check impacts on lock contention
  - slow down lock:contention-end (max: 10 msec)
  - right before it gets the lock
- Lock hold time tracking?
  - we may use delta between consecutive lock:contention-end for contended locks

System call tracing

- perf trace using syscall tables
  - ground work to support multiple ABI/platform
- Improved system call summary statistics using BPF
- BPF/BTF to read user pointers
- Now syscall tracing can read user pointers
  - should we get rid of the BPF augmentation?

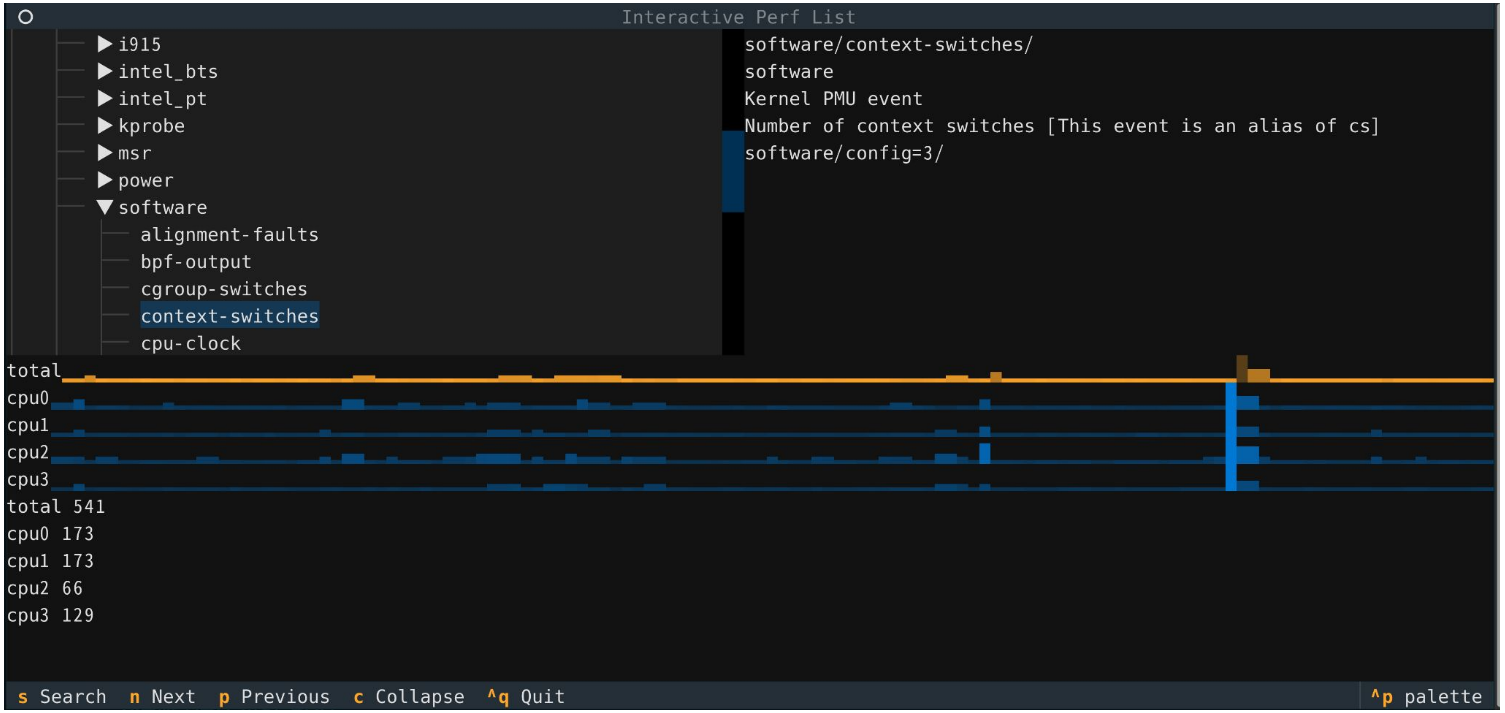
Guest machine profiling

- Can perf kvm record/report show guest applications?
- Scenario: whole machine (host + guests) profiling with trusted guests
  - can they shared ring buffers?
  - how to orchestrate guests?
  - how to guarantee atomicity while guest is writing…?
  - with mediated vPMU pass-through

Python support

- For first class python apps
    - easier to write, flexible, UIs
    - examples: twatch.py, ilist.py and more
- Perf script supports callback actions (for samples)
- New session APIs to handle raw event records directly

Python support



```
Interactive Perf List
    ▶ i915                                  software/context-switches/
    ▶ intel_bts                             software
    ▶ intel_pt                              Kernel PMU event
    ▶ kprobe                                Number of context switches [This event is an alias of cs]
    ▶ msr                                   software/config=3/
    ▶ power
    ▼ software
        alignment-faults
        bpf-output
        cgroup-switches
        context-switches
        cpu-clock
total
cpu0
cpu1
cpu2
cpu3
total 541
cpu0  173
cpu1  173
cpu2  66
cpu3  129

s Search  n Next  p Previous  c Collapse  ^q Quit                         ^p palette
```

東京 2025 LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025