Linux perf_events status update

Stephane Eranian
Google, Inc

Petascale Tools Workshop 2013

Agenda

The good, the bad, the ugly...

- new hardware support
- new perf_events kernel features
- new perf tool features
- new Gooda features
- Q&A

New hardware support

- Linux-3.10: Intel Ivy Bridge server (IvyTown, model 62)
  - core and uncore PMU (all boxes)
- Linux-3.11: Intel Haswell (desktop)
  - core, LBR, TSX, basic PEBS
- Linux-3.10: IBM Power 8

Haswell PMU new features

- TSX support
  - in_tx event filter: count event only when inside a transactional region
  - in_txcp event filter: do not count event in aborted transaction
  - TSX related events

- PEBS EventingIP
  - address of sampled instructions
  - eliminates off-by-1 skid (because IP captured at retirement)
  - off-by-1 IP still avail (branch sampling may require both)

- PEBS Data Linear Address (DLA)
  - capture data address for all PEBS memory events
  - can capture data address for specific cache events (loads/stores)

- LBR call-stack mode (cyclic taken branch buffer)
  - captures call instructions and pops last entry on return
  - enables callstack sampling with no frame-pointer, no debug info
  - does not work well with: leaf optimization, TSX aborts

Haswell PMU support

- TSX filters with perf:

```
$ perf stat -e cpu/cycles,in_tx=1/,cpu/cycles,in_tx=0/ noploop 2
noploop for 2 seconds
Performance counter stats for 'noploop 2':
              0 cpu/cycles,in_tx=1/
  7 370 922 986 cpu/cycles,in_tx=0/
    2,093117746 seconds time elapsed
```

- PEBS EventingIP
  - used with precise=2 (LBR not used anymore)

```
$ perf record -e cpu/event=0xc4,umask=0x2/pp noploop 2 (BR_INST_RETIRED:NEAR_CALL)
```

- PEBS Data Linear Address (DLA)
  - data address captured with many PEBS memory events
  - request DLA with PERF_SAMPLE_ADDR
  - regular PEBS Load Latency still available

```
$ perf record -d -e cpu/event=0xd0,umask=0x81/pp noploop 2 (MEM_UOPS_RETIRED:ALL_LOADS)
$ perf report -D | fgrep SAMPLE
PERF_RECORD_SAMPLE IP=0x401889 period: 286668 addr: 0x7f6f2474a3c0
```

Memory access sampling

- Available in Linux-3.10
  - requires HW support (NHM ld only, WSM, SNB, IVB, HSW)
  - PPC8 support in progress

- Samples load/store accesses
  - load: instr & data addr, instr latency, data source
  - store: instr & data addr, limited data source
  - data source abstracted: mem lvl, tlb lvl, snoop, lock
  - warning: instruction latency from dispatch (not just miss latency)

- perf tool support
  - `perf mem`: new wrapper command (record, report)
  - use `perf mem -D` for raw dump, easy to post-process

perf mem example

```
$ perf mem -t load rec test
$ perf mem -t load rep --stdio
# Samples: 23K of event 'cpu/mem-loads/pp'
# Total weight : 7394788
# Sort order   : local_weight,mem,sym,dso,symbol_daddr,dso_daddr,snoop,tlb,locked
#
# OV    Smpl   Weight Mem                        Sym       Obj           Data Sym        Data
# ..    ....   ...... ...    ......................   .......   ...............  ..........
 1.72%   92     1386 L3 hit [.] acquire.constprop.1  struct2  [.] object+0x18    struct2
 1.37%   73     1387 L3 hit [.] release.constprop.0  struct2  [.] object+0x18    struct2
 1.07%   57     1388 L3 hit [.] acquire.constprop.1  struct2  [.] object+0x18    struct2
 0.58%   31     1387 L3 hit [.] acquire.constprop.1  struct2  [.] object+0x18    struct2


$ perf mem -t load rep --sort=mem --stdio
# Samples: 23K of event 'cpu/mem-loads/pp'
# Total weight : 7394788
# Sort order   : mem
#
# Overhead       Samples             Memory access
# ........  ............  ........................
#
    97.95%          9915  L3 hit
     2.04%         13320  L1 hit
     0.01%            10  LFB hit
     0.00%             1  Local RAM hit
     0.00%             3  L2 hit
     0.00%             1  Uncached hit
```

hrtimer-based multiplexing

- available in Linux-3.11

- Multiplexing was piggybacked on timer ticks
  - tickless kernel: no timer tick when idle = no multiplexing
  - events may happen while core idle (think uncore events)

- add hrtimer per cpu for multiplexing
  - wake-up from idle to service timer
  - improved scaling accuracy for system-wide monitoring

- adjustable multiplexing rate per PMU instance via sysfs
  - default HZ, expressed in ms
  - Example: `echo 10 >/sys/devices/cpu/perf_event_mux_interval_ms`

Example: idle system, ref-cycles work on 1 counter only:

```
# perf stat -e ref-cycles,ref-cycles -a sleep 10
 Performance counter stats for 'sleep 10':
     5 825 973 800 ref-cycles    [50,01%]
     5 980 094 548 ref-cycles    [49,99%]
```

The bad: LateGO bug

- Local Memory Read / Load Retired events may undercount

```
MEM_LOAD_UOPS_RETIRED.LLC_HIT
MEM_LOAD_UOPS_RETIRED.LLC_MISS*
MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_MISS
MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HIT
MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HITM
MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_NONE
MEM_LOAD_UOPS_LLC_MISS_RETIRED.LOCAL_DRAM*
MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_DRAM*
MEM_TRANS_RETIRED.LOAD_LATENCY*
```

- Impacted CPU: SNB-EP (model 45)

- Workaround exists: very significant performance L3 latency increase
  - no kernel implementation
  - scripts do exist (Andi Kleen's latego.py script)

http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-e5-family-spec-update.pdf

The ugly: HT counter corruption

- measuring memory events may corrupt events on sibling thread

  ```
  MEM_LOAD_UOPS_RETIRED.*
  MEM_UOPS_RETIRED.*
  MEM_LOAD_UOPS_LLC_HIT_RETIRED.*
  MEM_LOAD_UOPS_LLC_MISS_RETIRED.*
  ```

  There may be more at-retirement events :-((

  ```
  Example:
  THREAD0: counter0=MEM_LOAD_UOPS_RETIRED:L3_MISS
  THREAD1: counter0 may be corrupted regardless of measured event
  ```

- impacted CPUs: SNB*, IVB*, HSW*

- no workaround in firmware
  - disable HT or measure only one thread/core (but clashes with NMI watchdog)

- Linux-3.11
  - blacklisting events on IVB even if HT is off (may add SNB, HSW soon)

- Google working on modifications to event scheduler
  - enforce mutual exclusion on sibling counters when corrupting events used

`perf stat` event grouping

- Available from Linux-3.8
- enforce event grouping from perf cmdline
  - events in group are always measured together
  - group cannot have more events than counters (ignoring constraints)
  - kernel support there since early days, no perf tool support

```
$ perf stat -e "{cycles,instructions}" noploop
        2324888687 cycles                   #    0.000 GHz
        2320675647 instructions             #    1.00  insns per cycle
```

But:

```
$ perf stat -e "{cycles,instructions,branches,branches,branches,branches}" 
noploop
        2324888687 cycles                   #    0.000 GHz
        2320675647 instructions             #    1.00  insns per cycle
        2319740061 branches
        2319740061 branches
   <not supported> branches
   <not supported> branches

Because of NMI watchdog using 1 counter.
```

perf record/annotate: event grouping

- make correlating events samples possible, at last!

```
$ perf record -e '{cycles,instructions}' noploop 2
$ perf report --group --stdio

# Samples: 16K of event 'anon group { cycles, instructions }'
# Event count (approx.): 9346466161
#
#          Overhead  Command      Shared Object                     Symbol
# ................  .......  .................  ........................
#
    99.95%  99.98%  noploop  noploop            [.] noploop
     0.02%   0.01%  noploop  [kernel.kallsyms]  [k] __slab_free
     0.01%   0.00%  noploop  ld-2.15.so         [.] _dl_relocate_object

$ perf annotate --group --stdio

Percent          |      Source code & Disassembly of noploop
-------------------------------------------------------------
                  :      0000000000400629 <noploop>:
    0.00    0.00 :        400629:       push   %rbp
    0.00    0.00 :        40062a:       mov    %rsp,%rbp
  100.00  100.00 :        40062d:       jmp    40062d <noploop+0x4>
```

More perf improvements

- `perf stat` interval printing

```
$ perf stat -a -I1000 -e cycles ...
#           time             counts event
     1.000102178      2,415,532,315 cycles
     2.000308349      2,414,348,054 cycles
```

- `perf stat` per-socket aggregation

```
$ perf stat -a -I1000 --per-socket -e cycles ...
#        time socket cpus             counts events
  1.000094565 S0        4         25,667,360 cycles
  2.000377213 S0        4         23,227,936 cycles
```

- `perf stat` per-core aggregation

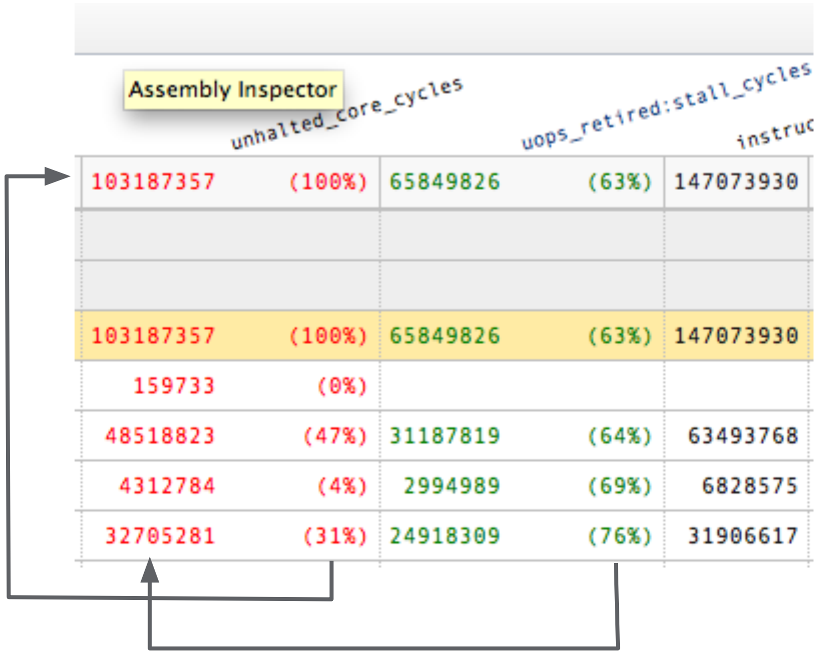
```
$ perf stat -a -I1000 --per-core -e cycles ...
  #        time core           cpus            counts events
  1.000100642 S0-C0             1           5,735,289 cycles
  1.000100642 S0-C1             1           4,257,992 cycles
  1.000100642 S0-C2             1           6,349,471 cycles
  1.000100642 S0-C3             1           6,312,706 cycles
```

Gooda updates

- Gooda analyzer
  - new support for ARMv7, PPC32, PPC64
  - Basic block execution counts using taken branch sampling
  - Diff utility creates report spreadsheets of differences with scaling
  - Sum & aggr utility creates report spreadsheets of sum/aggregation
  - Bug fixes

- Gooda collection scripts
  - use prime numbers for periods

- Gooda visualizer
  - hide columns
  - % cycles relative to total
  - % cycles relative to func or BB



unhalted_core_cycles		uops_retired:stall_cycles		instruc
103187357	(100%)	65849826	(63%)	147073930
103187357	(100%)	65849826	(63%)	147073930
159733	(0%)			
48518823	(47%)	31187819	(64%)	63493768
4312784	(4%)	2994989	(69%)	6828575
32705281	(31%)	24918309	(76%)	31906617

What's coming?

- more Haswell changes perf_event changes
  - lbr call-stack, lbr tsx extensions
  - perf tool TSX support
- sampling interrupted machine state
  - includes PEBS machine state
- value profiling support in perf tool
  - sampling values of function arguments (SNB and later)
- fix sample record header bug: no event identification
  - could not determine which event caused a sample record
  - all events had to request same meta-data (timestamp, cpu, ...)
  - wasted buffer and file space with LBR, for instance
  - add PERF_SAMPLE_IDENTIFIER sample_type

Conclusions

- good progress on perf_events kernel support
- good progress on tools
- cool new features on Haswell
- hardware bugs are killing us
  - need major validation effort across the community

PEBS memory events for Haswell DLA

PEBS Load events	PEBS store events
MEM_UOPS_RETIRED.STLB_MISS_LOADS	MEM_UOPS_RETIRED.STLB_MISS_STORES
MEM_UOPS_RETIRED.LOCK_LOADS	MEM_UOPS_RETIRED.LOCK_STORES
MEM_UOPS_RETIRED.SPLIT_LOADS	MEM_UOPS_RETIRED.SPLIT_STORES
MEM_UOPS_RETIRED.ALL_LOADS	MEM_UOPS_RETIRED.ALL_STORES
MEM_LOAD_UOPS_RETIRED.L1_HIT	MEM_LOAD_UOPS_RETIRED.L2_HIT
MEM_LOAD_UOPS_RETIRED.LLC_HIT	MEM_LOAD_UOPS_RETIRED.L1_MISS
MEM_LOAD_UOPS_RETIRED.L2_MISS	MEM_LOAD_UOPS_RETIRED.LLC_MISS
MEM_LOAD_UOPS_RETIRED.HIT_LFB	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_MISS
MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HIT	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HITM
UOPS_RETIRED.ALL	MEM_LOAD_UOPS_MISC_RETIRED.UC
MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_NONE	MEM_LOAD_UOPS_LLC_MISS_RETIRED.LOCAL_DRAM
MEM_LOAD_UOPS_LLC_MISS_RETIRED.LOCAL_DRAM_SNP_HIT	MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_DRAM
MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_DRAM_SNP_HIT	MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_HITM
MEM_LOAD_UOPS_LLC_MISS_RETIRED.REMOTE_FWD	MEM_LOAD_UOPS_MISC_RETIRED.NON_DRAM
MEM_LOAD_UOPS_MISC_RETIRED.LLC_MISS	

References

- Andi Kleen's pmu-tools
  - https://github.com/andikleen/pmu-tools
- LateGo
  - http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-e5-family-spec-update.pdf
- HT counter corruption
  - http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/3rd-gen-core-desktop-specification-update.pdf
- SNB-EP uncore PMU specification
  - http://www.intel.com/content/dam/www/public/us/en/documents/design-guides/xeon-e5-2600-uncore-guide.pdf
- Gooda tool
  - http://code.google.com/p/gooda