# perf_events status update

Stéphane Eranian
Google

CSCADS workshop
SnowBird, UT
June 2012

# Agenda

- Intel Sandy/Ivy Bridge support
- Taken branch sampling support
- perf and sysfs
- uncore PMU support
- PEBS Load Latency support
- perf tool update
- libpfm4 update
- WIP

Google™

# Supported HW

- AMD64
  - K8, Barcelona, Shanghai, Istanbul, Magny-Cours
  - Fam15h: Bulldozer (core PMU Only)
  - Fam14h (Bobcat), Fam12h
  -
- Intel X86
  - P6, Core Duo/Solo, Netburst (P4)(2.6.35)
  - Atom, Core, Nehalem-*/Westmere-*, SandyBridge-*, IvyBridge (3.5.0)
  - any processor with architected perfmon (PMU)

- ARM
  - ARMV6 (1136,1156,1176)
  - ARMV7 (cortex a5, a7,a8, a9, a15)

- IBM Power

- Alpha processors (EV67 and later)

Google™

# Intel SandyBridge

- Intel Sandy Bridge
  - 8 generic counters (4 with HT on)
  - full width counter writes (48-bit wrmsr)
  - PEBS Precise distribution (PDIR)
  - PEBS Precise store
  - PEBS Load Latency (LL) covers TLB and Lock
  - extended OFFCORE_RESPONSE events
  - LBR
  - uncore PMU

- PEBS bug => PEBS disabled
  - model 42, 45 impacted
  - need 6/6/12 ucode update (downloadcenter.intel.com)
  - need kernel fix (3.5.0) to check ucode version at runtime

# Support for Intel Sandy Bridge

- offcore_response: kernel 3.0.0
  - analyze memory traffic from a core's point of view
- PEBS Precise Dist (PDIR): kernel 2.6.39
  - mitigates the PEBS shadow effect
  - must be measured alone: too restrictive to be used
- LBR: kernel 3.4.0
  - full support via branch stack sampling abstraction
- uncore PMU (model 42, 45): kernel 3.5.0
  - counting mode
- PEBS Load Latency/Precise Store: under development
  - see later in the presentation

Google™

# Support for Intel Ivy Bridge

Intel Sandy Bridge 22nm shrink

- PMU identical
  - new events: MOVE_ELIMINATION, UOPS_EXECUTED
- PEBS PDIR
  - taken alone restriction lifted
- Sandy Bridge PEBS bug not present
- perf_events support in 3.5.0

Google™

# Taken branch sampling support (3.4.0)

- ability to sample N consecutive taken branches
  - per-event request
  - records: source, target, target prediction
  - ability to filter: any_call, any_ret, ind_call, any, u, k
- useful for call counts, BB execution counts, callgraph
- perf_events API abstraction: branch stack
  - `attr->sample_type |= PERF_SAMPLE_BRANCH_STACK`
  - `attr->branch_sample_type` = branch type filter mask
- implementation requires HW support
  - uses LBR on Intel X86
  - available on Core, NHM/SWM, SNB/IVB
  - recommend SNB and beyond due to bugs and limitations

Google™

# Branch sampling on Intel X86

- Last Branch Record (LBR)
  - capture control flow changes: br, intr, trap, fault, syscall
- Size
  - before NHM: 4 entries
  - since NHM: 16 entries
- Filtering
  - since NHM: branch type, priv levels
  - errata until SNB
  - priv level filter applies to branch target only
- kernel SW filtering
  - best effort
  - decode src insn to detect branch type

Google™

# perf tool support for branch sampling

- integrated with perf record, report
  - including report in TUI mode

```
main(){ while (l--) f1(l); }
f1(x){ if (x & 1) f2(); else f3(); }

$ perf record -j any_call -e br_inst_retired:call:u
$ perf report
 49.92% branchy2    [.] main    branchy2              [.] f1
 24.96% branchy2    [.] f1      branchy2              [.] f3
 24.96% branchy2    [.] f1      branchy2              [.] f2
```

- integration with precise sampling (:pp)
  - :pp use LBR to correct PEBS off-by-1 skid
  - if :pp LBR filter = user branch_stack filter
  - `perf record -b -e inst_retired:any:pp ...`

# PEBS memory access sampling

- PEBS-LL: load latency (NHM/WSM/SNB/IVB)
  - samples load accesses
  - collect: instr addr, data addr, latency, data src, TLB, lock
  - latency threshold filter
  - machine state at retirement of load
  - off-by-1 error on instr

- PEBS-ST: precise store (SNB/IVB only)
  - samples store accesses
  - collect: instr addr, data addr, TLB, L1D hit, lock
  - machine state at retirement of store
  - off-by-1 error on instr

# Memory sampling abstractions

- based loosely on initial patch by Lin Ming@Intel
- new generic hardware events:
  - `PERF_COUNT_HW_MEM_LOADS`
  - `PERF_COUNT_HW_MEM_STORES`
- latency threshold filter
  - attr->config1
- mem access infos requested via attr->sample_type:
  - ld/st addr: `PERF_SAMPLE_IP`
  - data addr: `PERF_SAMPLE_ADDR`
  - latency: `PERF_SAMPLE_LATENCY`
  - data src:`PERF_SAMPLE_DSRC`
- latency with PEBS-LL
  - core cycles from dispatch to globally observable
  - captures OOO execution:  large value does not always mean stalls

Google™

# perf tool support for mem access smpl

```
$ perf record -e mem_loads:u:precise=2 -l -d l1dhit
$ perf report
```

```
# Overhead  Lat  Mem      Symbol Shared Object  Data Addr      TLB access   Snoop   Locked
# ........  ...  ......  ............  ......  ............  ..........   ......  ......
#
    70.02%  319  L1 hit  [.] do_loop_s  l1dhit [.] 0x1a29680 L1 or L2 hit None    No
    29.10%  318  L1 hit  [.] do_loop_s  l1dhit [.] 0x1a29680 L1 or L2 hit None    No
     0.02%  409  L1 hit  [.] do_loop_s  l1dhit [.] 0x1a29680 L1 or L2 hit None    No
     0.02%  355  L1 hit  [.] do_loop_s  l1dhit [.] 0x1a29680 L1 or L2 hit None    No
     0.02%  350  L1 hit  [.] do_loop_s  l1dhit [.] 0x1a29680 L1 or L2 hit None    No
     0.02%  337  L1 hit  [.] do_loop_s  l1dhit [.] 0x1a29680 L1 or L2 hit None    No
```

```
$ perf report --sort=mem
```

```
#
# Overhead  Mem
# ........  ......
#
   100.00%  L1 hit
```

# Exporting PMU specs via sysfs

- 3.4.0: kernel exports via sysfs:
  - PMU config registers fields width and encodings layouts
  - PMU type => `attr->type`

- tools do not need to have arch spec knowledge to encode

```
/sys/devices/cpu$ ls
format power  rdpmc  subsystem  type  uevent
/sys/devices/cpu$ cat type
4
/sys/devices/cpu/format$ ls
any cmask edge event inv offcore_rsp pc umask
$ cat event
config:0-7
$ cat offcore_rsp
config1:0-63
```

# Intel uncore PMU support (3.5.0)

- written by Yan Zheng@Intel
- NHM/WSM: 1 uncore PMU, 8 counters, 1 fixed counter
- SNB/IVB: up to 14 uncore PMUs, 4 counters
  - n x C-Box, U-box, PCU, QPI, IMC, HA, PCIe
  - C-box, PCU filters support now included also
- monitoring of low level memory traffic
  - queues occupancy, PCIe traffic
  - correlating samples to cores impossible
- interrupt-based sampling has issues with C-states
  - counting mode only
  - use hrtimer to ensure 64-bit counter emulation

Google

# perf tool support for Intel uncore PMU

- SNB-EP : 8 cores = 8 C-Box (L3 slices)
  - uncore PMU types, reg layouts, common events via sysfs

```
/sys/devices# ls -d uncore*
uncore_cbox_0 uncore_cbox_3 uncore_cbox_6 uncore_imc_1 uncore_pcu   uncore_ha     uncore_r3qpi_1
uncore_cbox_1 uncore_cbox_4 uncore_cbox_7 uncore_imc_2 uncore_qpi_0 uncore_r2pcie
uncore_cbox_2 uncore_cbox_5 uncore_imc_0  uncore_imc_1 uncore_qpi_1 uncore_r3qpi_0

/sys/devices/uncore_qpi_0/format# ls
edge event inv thres umask

/sys/devices/uncore_qpi_0/format# cat umask
config:8-15

/sys/devices/uncore_qpi_0/events# ls
clockticks  drs_data  ncb_data txl_flits_active

/sys/devices/uncore_qpi_0/events# cat clockticks
event=0x14
```

- `perf stat:` use -C to avoid multiplexing

```
$ perf stat -a -C 0 -e uncore_qpi_0/clockticks/
$ perf stat -a -C 0 -e uncore_cbox0/event=0x37,umask=0x1/
```

Google™

# perf tool

- 3.4.0: branch stack sampling mode

- 3.4.0: new cmdline event parser
  - can name fields in counters, exported via sysfs
  - set per-event periods

```
perf stat -e cpu/event=0xc0,umask=0x1,inv/period=1000
```

- 3.4.0: event grouping with perf stat:
  - `perf stat --group -e a,b`
  - unfortunately only one group supported

- 3.0.0: `ref-cycles` generic event, at last!
  - not subject to freq scaling or Turbo mode
  - maps to `unhalted_ref_cycles` on Intel
  - kernel API update to encode fixed-counter only events

# libpfm4

- helper library to map event names to event encoding

- new in upcoming 4.3.0:
  - Intel Ivy Bridge support
  - Intel NHM/WSM/SNB/IVB uncore PMU
  - Arm A15
  - perf_events exclusive mode: excl
  - patch to integrate with perf: `perf stat --pfm-events ....`

- Warning on SNB-EP support
  - event table not yet published by Intel
  - using SNB event table as approximation

Google

# Work in progress

- capturing machine state register on interrupt
  - user level state (Redhat)
  - interrupted state (Google)
- capturing user level stack chunks on interrupt
  - for dwarf unwinding of user stack (Redhat)
- self-descriptive sample records (Google)
  - avoid collecting same info for all events

Google™

# Conclusion

- improved PMU HW support in kernel
- perf tool still needs a lot of improvements

Google™

# References

- Intel SNB uncore PMU
  - E5-2600 uncore performance monitoring guide
- Intel PMU specs May 2012
  - SDM Vol3b

Google™