Stéphane Eranian
Google, Inc

CSCADS workshop
Lake Tahoe, CA
August 2011
Agenda

- Intel Sandy Bridge support
- AMD Interlagos support
- Intel offcore_response support
- uncore PMU support
- cgroup monitoring support
- perf tool update
- libpfm4 update
Supported HW

- **AMD64**
  - K8, Barcelona, Shanghai, Istanbul (Magny-Cours?)
  - Fam15h: Bulldozer (core PMU)
  - Fam14h: Bobcat (AMD Fusion)

- **Intel X86**
  - P6, Core Duo/Solo, Netburst (P4)(2.6.35)
  - Atom, Core, Nehalem/Westmere, SandyBridge
  - any processor with architected perfmon (PMU)

- **ARM**
  - ARMV6 (1136,1156,1176)
  - ARMV7 (cortex-a8, cortex a9)

- **IBM Power**

- **Alpha processors** (EV67 and later)
New PMU Hardware

- 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
  - uncore PMU

- AMD Fam15h processor (Bulldozer)*
  - 6 core counters
  - Lightweight Profiling (LWP)
  - distinct uncore PMU
  - event scheduling constraints

* based on LKML patches posted by AMD (see http://lkml.org/lkml/2011/2/15/123)
Support for Intel Sandy Bridge

- support added in 2.6.39
  - generic event mappings
  - event scheduling
  - regular PEBS (incl LBR)

- offcore_response: limited support (2.6.39)

- PEBS-LL: patch under LKML review

- PEBS-ST: patch under LKML review

- PEBS-PDIR: partial support (2.6.39)

- uncore PMU: patch under LKML review
Support for AMD Fam15h (Bulldozer)

- core PMU support in 2.6.39
  - implements event scheduling constraints
  - mapping of generic events

- No LWP support yet
  - kernel xsave/xrstor patches from AMD: LKML review
  - rest of support can be encapsulated into user library

- no uncore PMU support yet
**PEBS memory access sampling**

- **PEBS-LL: load latency (NHM/WSM/SNB)**
  - samples location of load cache misses
  - must use event `MEM_TRANS_RETIRED LOADS`
  - collect: instr addr, data addr, latency, data src, L2TLB, lock
  - latency cycle filter (LD_LAT MSR)
  - machine state at retirement of load
  - still has off-by-1 error on instr

- **PEBS-ST: precise store (SNB only)**
  - samples location of store misses
  - must use event `MEM_TRANS_RETIRED STORES`
  - collect: instr addr, data addr, L2TLB hit, L1D hit, lock
  - machine state at retirement of store
  - still has off-by-1 error on instr
PEBS memory access sampling

- proposed perf_event abstractions (patch by Lin Ming @ Intel)

- new generic hardware events:
  - PERF_COUNT_HW_MEM_LOAD
  - PERF_COUNT_HW_MEM_STORE

- latency 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 via PERF_SAMPLE_EXTRA:
  - abstracted: MEM_LOAD_L1, MEM_LOAD_L2, MEM_LOAD_LOCAL...
● Lin's patch adds perf mem

$ perf mem -t load record make -j8
$ perf mem -t load report

Memory load operation statistics

<table>
<thead>
<tr>
<th>Category</th>
<th>Total Latency</th>
<th>Count</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1-local</td>
<td>28027</td>
<td>3355</td>
<td>8</td>
</tr>
<tr>
<td>L2-snoop</td>
<td>1430</td>
<td>29</td>
<td>49</td>
</tr>
<tr>
<td>L2-local</td>
<td>124</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>L3-snoop, found M</td>
<td>452</td>
<td>4</td>
<td>113</td>
</tr>
<tr>
<td>L3-snoop, found no M</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L3-snoop, no coherency actions</td>
<td>875</td>
<td>18</td>
<td>48</td>
</tr>
<tr>
<td>L3-miss, snoop, shared</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L3-miss, local, exclusive</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L3-miss, local, shared</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L3-miss, remote, exclusive</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L3-miss, remote, shared</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unknown L3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IO</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Uncached</td>
<td>464</td>
<td>30</td>
<td>15</td>
</tr>
</tbody>
</table>
Intel OFFCORE_RESPONSE event

- analyze memory traffic from core's point of view
  - can filter on type of memory request/response
  - core PMU event

- programming: config + counter + filter
  - filter uses extra MSR (shared when HT on in NHM/WSM)
  - filter programming: $2^{16}$ combinations (NHM/WSM), 37 bits on SNB!

- perf_events support
  - must manage HT shared registers
  - must encode event + filter

- abstracted offcore events
  - LKML: do not expose raw offcore, must abstract first!
  - WSM/NHM: subset mapped to generic cache events
• LL-read-access  
  ○ offcore_rsp:dmnd_data_rd:unc_hit:other_core_hit_snmp:other_core_hitm

• LL-read-miss  

• LL SNB mappings missing  
  ○ documentation issues mostly

• no RAW access

• not good enough for NUMA breakdown  
  ○ need local vs. remote traffic  
  ○ patch proposed by Zijlstra with new generic cache events
Support for SNB PDIR sampling

- **PEBS skid (applies to any PEBS event)**
  - N cycles of shadow when PEBS is armed
  - Z instructions may retire during that window
  - May bias the sample distribution
  - Period P \(\Rightarrow\) effective period = \(P + Z\)
  - Z varies depending on executed code

- **Precise Distribution Instruction Retired (PDIR)**
  - Mitigates the PEBS skid
  - Works only when sampling retired instructions
  - Must use `INST_RETIRED:PREC_DIST` event

- **perf_event support:**
  - Users must ensure only event on PMU
  - Should set `attr->exclusive` \(\Rightarrow\) clash with NMI watchdog
Generic stall events

- two new generic PMU events:
  - `PERF_COUNT_HW_STALLED_CYCLES_FRONTEND`
  - `PERF_COUNT_HW_STALLED_CYCLES_BACKEND`
  - no clear definitions

- LKML: definitions not needed, high correlation good enough
  - counts do not have to be precise
  - but: how do you associate a cost then?
Multiple PMUs support

- major code restructuring in 2.6.37
  - manage multiple distinct PMUs simultaneously
  - events inside group must be from the same PMU

- required for uncore PMUs support

- how to identify PMU to encode event?
  - each PMU => unique id in attr->type
  - use sysfs, e.g., uncore:
    /sys/bus/event_source/devices/uncore/type
    /sys/bus/event_source/devices/ibs_op/type

- LKML: extend sysfs to expose basic event encoding
  - /sys/bus/event_source/devices/uncore/events/cycles
  - usage: perf stat -a -C 0 -e uncore:cycles
Intel uncore PMU support

- NHM/WSM
  - 1 fixed counter (uncore clock ticks)
  - 8 generic counters (44 bits)

- Monitoring of very low level memory traffic
  - Sample correlation to core or program impossible

- Interrupt-based sampling has issues with C-states
  - Uncore PMU interrupt not delivered to C-sleeping cores
  - Cannot lose interrupt => counter wraps around

- Lin Ming @Intel patch provides counting only
  - Uses hrtimer to avoid wrap around counting issues
  - Raw mode access for uncore events allowed
Per-container monitoring

• resource container (cgroup)
  ○ cpuset, mem, scheduling

• can now monitor execution inside a specific cgroup:
  ○ **ex:** perf stat -a -e cycles -G foo -- sleep 1

• extension of system-wide
  ○ on each cpu, monitoring is active only when running a thread that belongs to the monitored cgroups

• cgroup specified via fd from cgroupfs:
  ○ fdc = open("/dev/cgroup/foo", O_RDONLY)
  ○ fd = perf_event_open(&attr, fdc,0,PERF_FLAG_PID_CGROUP, 0)

• upstream since 2.6.39 (contributed by Google)
Still missing

- taken branch sampling
  - leverage LBR on Intel

- Intel NHM/WSM/SNB uncore (desktop SKUs)
  - patch under review

- AMD IBS (AMD contributing)
  - patches just posted by Robert Richter @ AMD
  - patch looks very good, should get in quickly

- ability to capture machine state (regs) on interrupt

- ability to control multiplexing rate (use hrtimers)

- improved event scheduling (maximize PMU usage)

- Intel X86: unhalted_reference_cycles event
perf tool

- Curses-based GUI (NEWT toolkit)
  - not very useful

- cgroup support
  - perf stat -e cycles -G foo -a -- sleep 1

- perf report --symfs
  - point to dir with unstripped
libpfm4

- helper library to map event names to event encoding

- libpfm4-1.0 released
  - git repository: perfmon2.sf.net

- restructured code to support OS API attributes:
  - ex: INST_RETIRED:period=2000000:c=1:i

- memory usage improvements
  - 37% size reduction on Intel X86 memory usage

- more HW support
  - all X86 processors, IBM Power, SPARC, ARM Cortex A8/A9

- patch for perf tool posted but still not integrated by maintainer
Conclusion

- improved PMU HW support in kernel
- lots of kernel patches coming to close the gaps
- perf tool still needs a lot of improvements