Linux perf_events status update

Stephane Eranian
Google

Petascale Tools Workshop 2015
Agenda

- New hardware support
- New kernel features
- Exploiting the uncore PMU on Intel servers
- Q&A
New hardware support

- Intel Haswell server (HSX) uncore in Linux v3.18
  - memory controller, power, qpi, pcie, …

- Intel Broadwell client (BDW, Xeon D) in Linux v4.1
  - core PMU, power (RAPL), memory controller (IMC)

- Intel Broadwell SoC (Xeon D) uncore in Linux v4.2
  - memory controller, power, pcie

- Intel SkyLake client in Linux v4.3
  - core PMU, includes new LBR, PEBS features
SkyLake new features

- Last Branch Buffer (LBR) has 32 entries (2x Haswell)
- Timed LBR: basic block cycle duration
  - capture cycle duration between consecutive branches
  - LBR record: 3x uint64_t now (50% increase)
- TSC is captured by PEBS
- PEBS precise Front-End sampling
  - sample where I-TLB, I-CACHE misses occur
- Patches by Andi Kleen (Intel) posted on LKML for Linux v4.3
LBR Call stack mode

- LBR records call branches and pops the last entry on return (Haswell)
  - not work in certain corner cases; leaf call optimization

- Available in Linux v3.19
  - advantage: no framepointer, no dwarf needed, no user regs/stack snapshots
  - gotcha: only work in user mode (hw bug)
  - new `PERF_SAMPLE_BRANCH_CALL_STACK` branch sample type

- perf tool integration
  - `perf record --call-graph lbr -e cycles:uk... ⇒ user = lbr, kernel = FP`
  - `perf record --call-graph lbr -e cycles:k ... ⇒ error`
  - `perf record --call-graph lbr -e cycles:u ... ⇒ lbr callstack`
  - reporting: `perf report` and navigate the callstacks
Configurable Timestamp clock source

- Can configure the timestamp clock source per-event (Linux v4.0)
  - synchronize with user level generated samples from runtimes
  - was using kernel internal-only clock-source (`sched_clock()`)

- `perf_event_attr.clockid = N, .use_clockid = 1`
  - N is a POSIX clock identifier (MONOTONIC, REALTIME, RAW, ...)

- Example: correlate with Java JVMTI JIT information
  - JVMTI agent uses `clock_gettime(CLOCK_MONOTONIC)`
  - `perf_event_attr.clockid = CLOCK_MONOTONIC`
  - jit compiler events correlate automatically with perf samples
Sampling interrupt machine state

- Capture register state at PMU interrupt (Linux v3.19)
  - can specify which registers to capture per event

- What is that useful for?
  - sampling value of registers at particular points
  - example: Am I calling `memset()` mostly with a size of 16?

- Value Profiling: sample values of function arguments
  - requires: reg-based calling convention (x86_64, ppc64, ...)
  - Intel x86: sample call instructions at target (1st instr of func) and save regs
  - Intel x86: use `br_inst_retired:near_call + pebs + skid`
    - $ perf record -I -e cpu/event=0xc4,umask=0x2/p
  - visualization: `perf report -D` (for now),
Monitoring L3 cache occupancy

- Intel Cache Monitoring Technology (CMT)
  - Xeon specific feature, available on Haswell server
  - monitor L3 cache occupancy per process

- Available in Linux v4.2
  - can operate in per-thread and per-cpu mode incl. containers (cgroup)
  - new PMU: intel_cqm, new event: llc_occupancy
    - perf stat -I 1000 -e intel_cqm/llc_occupancy/ my_program

- Cache Allocation Technology (CAT)
  - enforce limits on L3 cache space (ways) available
  - patches posted on LKML by Intel
Cache monitoring examples (Haswell server)

$ perf stat -e intel_cqm/llc_occupancy/ -I 1000 ./triad
#       time       counts  unit     events
1.003202964 47185920.00 Bytes intel_cqm/llc_occupancy/
2.006316523 47480832.00 Bytes intel_cqm/llc_occupancy/

$ taskset -c 0 triad & taskset -c 18 triad &
$ # perf stat -a -e intel_cqm/llc_occupancy/ -I 1000 sleep 100
#       time       counts  unit     events
1.003116711 94371840.00 Bytes intel_cqm/llc_occupancy/
2.006269988 94371840.00 Bytes intel_cqm/llc_occupancy/
TSC, APERF, MPERF exposed!

- Provide a way to add free-running counters support
  - free-running: non-stop, no-interrupt, fixed register

- Patch adds TSC, APERF, MPERF
  - APERF: increments in proportion to actual performance
  - MPERF: increments in proportion to a fixed frequency
  - ratio APERF/MPERF architecturally defined

- New `freq` PMU with new events: `tsc`, `aperf`, `mperf`
  - no sampling, no vmm
  
  $ perf stat -a freq/tsc/,freq/aperf/,freq/mperf/ -I 1000 sleep 10

- Just a proposal on LKML (Intel, Andy Lutomirski)
Exploiting uncore PMUs better in servers

- Intel Xeon server have lots of PMUs

- Can monitor I/O, memory, power, inter-socket comm

- Each PMU has generic counters (+ some fixed)

- Only support system-wide measurements

- No sampling mode in perf_events
  - no interrupt (oftentimes)
  - shared resources: cannot identify core
  - only sees physical addresses

- Kernel releases with support
PCIe bandwidth (Intel IvyTown)

- L3 coherency agent PMU (**Cbox**) (**uncore_cbox_***)  
  - one Cbox agent per physical core  
  - use TOR_INSERTS event + opcode match PCIe opcodes

```
$ perf guncore -M pcie_bw
#--------------------------------------------------------------
#           Socket0            |           Socket1            |
#--------------------------------------------------------------
#        PCIe Bandwidth        |        PCIe Bandwidth        |
#  PCIe->RAM,QPI  RAM,QPI->PCIe|  PCIe->RAM,QPI  RAM,QPI->PCIe|
#           MB/s           MB/s|           MB/s           MB/s|
#--------------------------------------------------------------
  148.20           3.61            0.00           0.00
  139.24           8.78            0.00           0.00
  132.66           4.13            0.00           0.00
```
Memory bandwidth (IvyTown)

- Integrated Memory Controller (IMC) PMU (uncore_imc_*)
  - CAS_COUNT event to break down reads vs. write

- per-socket view useful to detect imbalance

```bash
$ perf guncore -M mem_bw
```

<table>
<thead>
<tr>
<th>Socket0</th>
<th>Socket1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Bandwidth</td>
<td>RAM Bandwidth</td>
</tr>
<tr>
<td>Wr Mb/s</td>
<td>Rd Mb/s</td>
</tr>
<tr>
<td>5.83</td>
<td>24.27</td>
</tr>
<tr>
<td>6.42</td>
<td>20.09</td>
</tr>
</tbody>
</table>
QuickPath Interconnect bandwidth

- QPI PMU (uncore_qpi_*)
  - RXL_FLITS and TXL_FLITS events
- detect remote socket accesses
- detect workload imbalance

```bash
$ perf guncore -M qpi_bw
```

<table>
<thead>
<tr>
<th></th>
<th>Socket0</th>
<th></th>
<th>Socket1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>QPI Bandwidth</td>
<td></td>
<td>QPI Bandwidth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM,PCIe-&gt;QPI</td>
<td>MB/s</td>
<td>RAM,PCIe-&gt;QPI</td>
<td>MB/s</td>
<td></td>
</tr>
<tr>
<td>QPI-&gt;RAM,PCIe</td>
<td>MB/s</td>
<td>QPI-&gt;RAM,PCIe</td>
<td>MB/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.79</td>
<td>5.31</td>
<td>5.31</td>
<td>9.79</td>
</tr>
<tr>
<td></td>
<td>16.16</td>
<td>15.66</td>
<td>15.65</td>
<td>16.17</td>
</tr>
</tbody>
</table>
C-state monitoring

- Power Controller Unit (PCU) PMU (uncore_pcu)
  - POWER_STATE_OCCUPANCY event
- useful to detect core utilization
- power saving opportunities

```bash
$ perf guncore -M cstate
```

<table>
<thead>
<tr>
<th></th>
<th>Socket0</th>
<th></th>
<th>Socket1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C-states</strong></td>
<td></td>
<td><strong>C-states</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cores</strong></td>
<td><strong>Cores</strong></td>
<td><strong>Cores</strong></td>
<td><strong>Cores</strong></td>
<td><strong>Cores</strong></td>
</tr>
<tr>
<td>in C0/C1</td>
<td>in C3</td>
<td>in C6/C7</td>
<td>in C0/C1</td>
<td>in C3</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>0.24</td>
<td>0.00</td>
<td>11.76</td>
<td>0.20</td>
<td>0.00</td>
</tr>
<tr>
<td>0.24</td>
<td>0.00</td>
<td>11.76</td>
<td>0.18</td>
<td>0.00</td>
</tr>
</tbody>
</table>
uncore view

- combining metrics to get a global view
  - question: Am I accessing remote memory?
  - perf guncore tool: `mem_bw + qpi_bw`

```
$ perf guncore -M mem_bw,qpi_bw
```

<table>
<thead>
<tr>
<th>#</th>
<th>RAM Bandwidth</th>
<th>QPI Bandwidth</th>
<th>RAM Bandwidth</th>
<th>QPI Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Wr MB/s</td>
<td>Rd MB/s</td>
<td>RAM,PCIe-&gt;QPI MB/s</td>
<td>QPI-&gt;RAM,PCIe MB/s</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>1</td>
<td>6.75</td>
<td>19.40</td>
<td>6.57</td>
<td>4.85</td>
</tr>
<tr>
<td>2</td>
<td>5.91</td>
<td>21.69</td>
<td>9.71</td>
<td>3.43</td>
</tr>
<tr>
<td>3</td>
<td>4.46</td>
<td>17.14</td>
<td>5.68</td>
<td>2.29</td>
</tr>
</tbody>
</table>

- Many more metrics possible, consult uncore programming guide
Intel Processor Tracing (PT)

- Hardware tracing support introduced with Broadwell processors
  - can trace control flow change in a compressed trace format

- kernel support via perf_events interface (Linux v4.1)
  - a lot of extensions to the sampling buffer (auxiliary buffer)
  - appears as new PMU: intel_pt

- perf tool support not quite complete in Linux 4.2
  - perf record -e intel_pt///u
  - perf report
Miscellaneous progress

- SandyBridge, IvyBridge, Haswell Hyperthreading counter corruption bug workaround
  - cross HT counter corruption with events 0xd0, 0xd1, 0xd2
  - sophisticated kernel workaround developed by Google
  - patch integrated into v4.1 (fixed in 4.2)

- perf JIT code profiling support
  - vastly benefit from the per-event clock source support
  - rebased to 3.19
  - still not merged in as of 4.3
  - needs some more cleanups based on LKML feedback

- IBM pushing Power8 Nest (uncore) support on LKML
  - Link, Memory bandwidth
  - Power
Conclusions

● Good progress this year
  ○ better set of features
  ○ stabilization and bug fixes

● SkyLake PMU looks very good

● Intel Cache Occupancy Monitoring is in

● Uncore PMU provides a wealth of useful information

● Intel Processor Trace is coming very soon now
References

- Intel official event tables
  - [https://download.01.org/perfmon/](https://download.01.org/perfmon/)

- Intel Cache Monitoring & Cache Allocation Technologies
  - [CAT patch on LKML](https://lkml.org/lkml/2018/4/22/1160)

- TSC/APERF/MPERF patch form LKML

- Intel Processor Trace (PT) support (Linux 4.1 + Broadwell processor)
  - Intel contribution (Adrian Hunter, Andi Kleen, Alexander Shishkin)
  - until fully merged, needs custom perf tool available on GitHub [here](https://github.com/akleinen/)

- Intel uncore PMU guides
  - links to all guides available [here](https://ark.intel.com)

- IBM Nest patches