AMD IBS and Northbridge Counters in perf

Robert Richter <rric@kernel.org>

2nd CERN Advanced Performance Tuning workshop
November 21, 2013

Content

- Performance Counters
- Instruction Based Sampling (IBS)

Performance Counters, Overview

	Introduced	Note
x86 Performance Counters	K7 10h/00h	northbridge events introduced with family 10h (only one counter per node)
Core Performance Counters	15h/00h-0Fh Bulldozer (BD)	6 architectural defined core counters, but event constraints
Northbridge Performance counters, Counters	15h/00h-0Fh Bulldozer (BD)	4 architectural defined nb per-node msrs, no constraints
Update: Performance Ctr	15h/02h BD Gen 2	New events introduced
L2I performance counters	16h Jaguar	4 counters for L2 cache specific events (i.e. L2 cache misses)

Core Performance Counters

- introduced with cpu family 15h
- 6 counters residing in a new msr range
- separated from northbridge counters
- CPUID detection
- Counter constraints, certain events may only be schedule on certain counters
- Linux kernel support since v2.6.39
- See commit id 4979d2729af22f6ce8faa325fc60a85a2c2daa02 http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=commitdiff;h=4979d2729af22f6ce8faa325fc60a85a2c2daa02

Family 10h Northbridge Performance Counters

- 4 counters (same counters as for other performance events)
- Only one northbridge event per node and counter may be used
- Implemented by Stephane Eranian
- Linux kernel support since v2.6.34
- See commit id 38331f62c20456454eed9ebea2525f072c6f1d2e
  http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=commitdiff;h=38331f62c20456454eed9ebea2525f072c6f1d2e

Northbridge Performance Counters

- Introduced with Family 15h
- 4 counters (separated from core performance counters)
- CPUID detection
- Implemented as separated PMU
- perf record event selection (-e uncore/foo=bar/)
- upstream since v3.10

Northbridge Performance Counters, Restrictions

- Per-node MSRs (changes on one cpu are visible on another cpu of the same node)
- Only one northbridge event per node and counter may be used
- Interrupt delivery to all cores of the node
- Counting modes not supported (event selection MSR modified compared to core counters):
  - Host/Guest Only
  - Counter Mask
  - Invert Comparison
  - Edge Detect
  - Operating-System Mode
  - User Mode

L2I Performance Counters

- count specific L2 events that occur in a core of the compute unit
- count the activity of all cores of a compute unit
- 4 new counters, similar to northbridge counters
- upstream since v3.10
- See BKDG for Family 16h:
  http://support.amd.com/TechDocs/48751_16h_bkdg.pdf

Instruction Based Sampling (IBS), Overview

	Introduced	**Note**
IBS	10h/00h Barcelona	
Update: IBS	10h/02h Shanghai	IBS CPUID detection, micro-op counting mode for IBS op, several small changes
Update: IBS	12h/00h Llano	IBS enhancements: RipInvalidChk and OpCntExt

Instruction Based Sampling (IBS), Introduction

- Hardware profiling mechanism
- Selects a random instruction fetch or micro-op after certain number of clock cycles or retiered micro-ops
- Records specific performance information about the operation
- ibs_fetch and ibs_op can be used separate
- Both have a control register and a number of register with collected data (exact rip or data address, several pseudo event can be derived)
- Documentation:
  - latest BKDG: http://developer.amd.com/wordpress/media/2012/10/42301_15h_Mod_00h-0Fh_BKDG1.pdf
  - latest APM vol 2: http://developer.amd.com/wordpress/media/2012/10/24593_APM_v21.pdf

IBS Features in the kernel

- Upstream since v3.5 (kernel), v3.7 (sysfs tool support)
- Registration of two PMUs in the kernel (ibs_op/ibs_fetch)
- Precise-event sampling of event 76h (cycle counting) and C1h (uops retired)
- full support of the perf_event_open() syscall
- Raw data sampling to pass the IBS register contents to userland
- Perl script support for data post-processing
- Generic perf tool support (perf report/record/top/script)
- sysfs support for event selection
- Support of OpCntExt
- pmu/type mapping for perf.data post processing

Using IBS with perf

- Precise-event sampling uses IBS for AMD CPUs
- IBS is used if the precise modifier is set (:p)
- Collect and process IBS samples:

```
# perf record -a -e cpu-cycles:p ...    # use ibs op counting cycle count
# perf record -a -e r076:p ...          # same as -e cpu-cycles:p
# perf record -a -e r0C1:p ...          # use ibs op counting micro-ops
```

- The counting mode (cycle/micro-op counting) is selected depending on the selected event (76h/C1h)
- Example:

```
# perf record -R -a -e r076:p -c $((0x1FFFE0)) <workload>
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 1.234 MB perf.data (~53932 samples) ]
```

Using IBS with perf-record (2)

- Add sysfs format entries for AMD IBS PMUs:

```
# find /sys/bus/event_source/devices/ibs_*/format
/sys/bus/event_source/devices/ibs_fetch/format
/sys/bus/event_source/devices/ibs_fetch/format/rand_en
/sys/bus/event_source/devices/ibs_op/format
/sys/bus/event_source/devices/ibs_op/format/cnt_ctl
```

- This allows to specify following IBS options:

```
$ perf record -e ibs_fetch/rand_en=1/GH …
$ perf record -e ibs_op/cnt_ctl=1/GH ...
```

Note: cnt_ctl only AMD family 10h RevC and above

IBS data sample post-processing with perf-script (1)

- Generate:

```
# perf script -g perl > /dev/null
generated Perl script: perf-script.pl
```

- ... and then modify perf-script.pl:

```
# perf script event handlers, generated by perf script -g perl
# Licensed under the terms of the GNU GPL License version 2|
use lib "$ENV{'PERF_EXEC_PATH'}/scripts/perl/Perf-Trace-Util/lib";
use lib "./Perf-Trace-Util/lib";
use Perf::Trace::Core;
use Perf::Trace::Context;
use Perf::Trace::Util;

sub process_event
{
       my ($raw_data) = $_[3];
       my ($caps, @raw_data) = unpack("LQ*", $raw_data);

       print((join ", ", sprintf("0x%08X", $caps),
               map { sprintf("0x%016X", $_) } @raw_data),
              "\n");
}
```

IBS data sample post-processing with perf-script (2)

- Post-process samples:

```
# perf script -s perf-script-ibs.pl | head
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF8100873F, 0x0000000000040004, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF8100873F, 0x0000000000040004, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF810DD309, 0x0000000000290023, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF8100873F, 0x0000000000040004, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF8100873F, 0x0000000000040004, ...
0x00000007, 0x000000000006FFFF, 0x00007F8AA11E3F68, 0x0000000000070004, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF8100873F, 0x0000000000040004, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF8100873F, 0x0000000000040004, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF811AEC9E, 0x0000002800090002, ...
0x00000007, 0x000000000006FFFF, 0xFFFFFFFF811AECE9, 0x00000000000A0002, ...

...
```

IBS setup with perf_event_open() syscall

- Preparing the syscall is easy:

```
memset(&attr, 0, sizeof(attr));
attr.type = type;
attr.sample_type   = PERF_SAMPLE_CPU | PERF_SAMPLE_RAW;
attr.sample_period = config->sample_period;
attr.config        = config->config;
```

- But it ends up in a complex application, you need to:
  - call multiple syscalls depending on the context and number of events and cpus,
  - start your application as child and control it with ptrace,
  - watch file descriptors and read out mmaped buffers
  - process each sample.
- Full example code, see here: https://lkml.org/lkml/2011/9/7/204 (Note: not updated to current mainline)

Precise event sampling with IBS for AMD CPUs

Skiddy '-e cpu-cycles' versus skid-less '-e cpu-cycles:p' output:

```
# perf annotate -k vmlinux -s _raw_spin_lock_irqsave -i perf-r076.data | cat
# perf annotate -k vmlinux -s _raw_spin_lock_irqsave -i perf-r076p.data | cat

 Percent |  Percent |     Source code & Disassembly of vmlinux
------------------------------------------------
  cpu-cycles        :
    |    : cpu-cycles:p   Disassembly of section .text:
    |    :     |    :
    v    :     v    :     ffffffff8145036a <_raw_spin_lock_irqsave>:
    0.00 :     0.00 :     ffffffff8145036a:       push   %rbp
    0.00 :     0.00 :     ffffffff8145036b:       mov    %rsp,%rbp
    0.00 :     0.00 :     ffffffff8145036e:       callq  ffffffff81456c40 <mcount>
    0.00 :     0.00 :     ffffffff81450373:       pushfq
    0.00 :     0.00 :     ffffffff81450374:       pop    %rax
    0.00 :     0.00 :     ffffffff81450375:       cli
    0.00 :     0.00 :     ffffffff81450376:       mov    $0x100,%edx
    0.00 :     0.00 :     ffffffff8145037b:       lock xadd %dx,(%rdi)
    0.00 :     2.78 :     ffffffff81450380:       mov    %dl,%cl
    0.00 :     0.00 :     ffffffff81450382:       shr    $0x8,%dx
   10.34 :     2.78 :     ffffffff81450386:       cmp    %dl,%cl
    0.00 :    11.11 :     ffffffff81450388:       je     ffffffff81450390 <_raw_spin_lock_irqsave+0x26>
   10.34 :    72.22 :     ffffffff8145038a:       pause
   65.52 :     2.78 :     ffffffff8145038c:       mov    (%rdi),%cl
   13.79 :     8.33 :     ffffffff8145038e:       jmp    ffffffff81450386 <_raw_spin_lock_irqsave+0x1c>
    0.00 :     0.00 :     ffffffff81450390:       leaveq
    0.00 :     0.00 :     ffffffff81450391:       retq
```

AMD IBS and Northbridge Counters in perf

Questions?