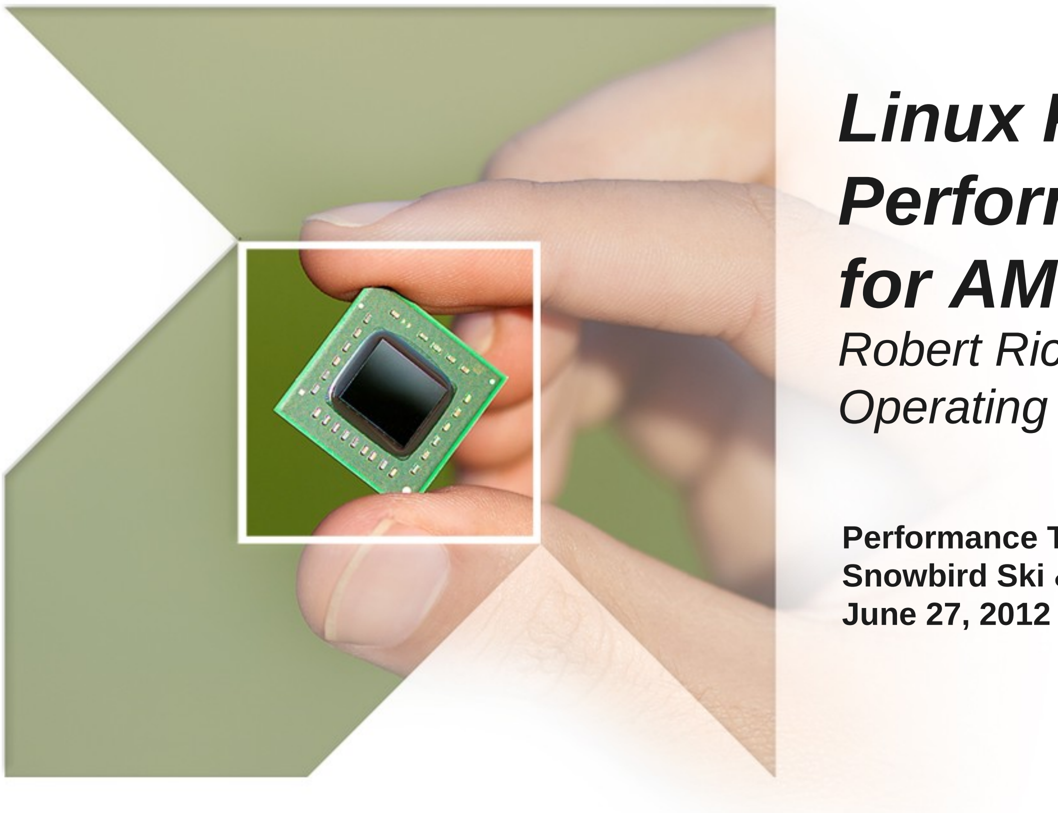

Linux Kernel and Performance Tools support for AMD PMUs

Robert Richter <robert.richter@amd.com>
Operating System Research Center

Performance Tools for Extreme-scale Computing
Snowbird Ski & Summer Resort, Snowbird, Utah, USA
June 27, 2012

Content

- Software Enablement for AMD CPUs
- Performance Events Subsystem
- Brief Introduction to perf
- Overview of PMU Features in current CPUS
- Performance Counter Support
- Instruction Based Sampling (IBS)
- Lightweight profiling (LWP)

AMD Software Enablement of PMU features

- Linux upstream and stable kernels
- Userland and library support:
  - perf tools
  - oprofile (maintenance mode)
  - libpfm
  - papi
- Distro support:
  - RHEL 5, 6, 7 (means also support for CentOS and Scientific Linux)
  - Suse SLES 11 SP1/SP2, SLES 12
- Additional software enablement support depending on customer request

Performance Events Subsystem (1)

- Most important Linux kernel profiling subsystem
- Many developers contributing code to it
- Other performance kernel subsystems (esp. oprofile) are deprecated and only in maintenance mode (no further feature implementation)
- Kernel and userland:
  - perf_event_open syscall
  - perf - Performance analysis tools for Linux
- Both parts reside in the Linux kernel tree

Performance Events Subsystem (2)

- Maintainers:
  - Peter Zijlstra <a.p.zijlstra@chello.nl>
  - Paul Mackerras <paulus@samba.org>
  - Ingo Molnar <mingo@elte.hu>
  - Arnaldo Carvalho de Melo <acme@ghostprotocols.net>
- Files:
  - kernel/events/*
  - include/linux/perf_event.h
  - arch/*/kernel/perf_event*.c
  - arch/*/kernel/*/perf_event*.c
  - arch/*/kernel/*/*/perf_event*.c
  - arch/*/include/asm/perf_event.h
  - arch/*/lib/perf_event*.c
  - arch/*/kernel/perf_callchain.c
  - tools/perf/

Performance Events Subsystem, Documentation (1)

- Bad but enough documentation (there is the source code):
  - http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=summary
- Vince Weaver's Unofficial Linux Perf Events Web-Page:
  - http://web.eecs.utk.edu/~vweaver1/projects/perf-events/index.html
  - http://web.eecs.utk.edu/~vweaver1/projects/perf-events/programming.html (syscall programming)
- Kernel Documentation (sometimes outdated)
  - Design: http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=tools/p
  - Examples: http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=tools/p

Performance Events Subsystem, Documentation (2)

- perf man pages (man perf etc.)
- LWN Articles (see also Performance monitoring on http://lwn.net/Kernel/Index/
  - Dueling performance monitors, http://lwn.net/Articles/310260/
  - Followups: performance counters, ksplice, and fsnotify, http://lwn.net/Articles/311850/
  - Perfcounters added to the mainline, http://lwn.net/Articles/339361/
  - Fun with tracepoints, http://lwn.net/Articles/346470/
  - KS2009: The future of perf events, http://lwn.net/Articles/357481/
  - perf trace: support for general-purpose scripting, http://lwn.net/Articles/355622/

Important perf tools

```
perf stat    # run command and gather a counter statistic from it
perf top     # generate and display a events in realtime
perf record  # collect sampling data
perf report  # analyse and display sampling data
perf list    # list available events on the system
```

See man pages of perf-stat(1), perf-top(1), perf-record(1), perf-report(1), perf-list(1)

PMU Features in current CPUs, Overview

	Introduced	Note
Performance Counters:		
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	15h/00h-0Fh Bulldozer (BD)	4 architectural defined nb counters, per-node msrs, no constraints
Update: Performance Counters	15h/02h next-gen BD	New events introduced
Instruction Based Sampling:		
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
Lightweight Profiling:		
LWP	15h/00h-0Fh Bulldozer (BD)	

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=4979d

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=38331

Northbridge Performance Counters

- Introduced with Family 15h
- 4 counters (separated from core performance counters)
- CPUID detection
- Patches send to LKML in June 2012, see https://lkml.org/lkml/2012/6/19/324
- patches haven't been accepted and are not yet upstream
- Intel's uncore patches were applied
- Implemented as separated PMU
- Will rework and repost an uncore-like implementation in the next weeks
- perf record event selection (-e uncore/foo=bar/)

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

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)

IBS Pseudo Events

- From an IBS sample pseudo events can be derived:
  - IBS fetch data (e.g. L1I hits/misses)
  - Events to Measure All, Branch, Return and Resync Ops (e.g. branch predictor)
  - Events for Ops that Perform Load and/or Store Operations (e.g. L1D/L2D hits/misses)
  - IBS Northbridge Derived Events (e.g. locale or remote nb access)
- See Software Optimization Guides for Family 15h for a list of events: http://support.amd.com/us/Processor_TechDocs/47414_15h_sw_opt_guide.p

IBS in the Linux kernel

- Supported for a long time by oprofile
- perf kernel support upstream in v3.5 (let's celebrate!)
- perf tool support still pending
- Latest off-tree perf tools patches:
  http://git.kernel.org/?p=linux/kernel/git/rric/oprofile.git;a=shortlog;h=refs/heads

IBS Features, Upstream

- 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
- System-wide mode (per-thread monitoring not yet implemented)
- Perl script support for data post-processing
- Generic perf tool support (perf report/record/top/script)

IBS Features, Outlook

- Per-thread monitoring
- sysfs support for event selection
- Counting mode (perf-stat)
- Support of OpCntExt
- Pseudo event support (userland-side sample filtering)
- pmu/type mapping for perf.data post processing

Using IBS with perf-record

- 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) ]
```

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
```

Lightweight Profiling (LWP), Introduction

- Very low overhead
- Allows user mode processes to profile themselves
- Hardware writes data samples into process memory
- Userland controls profiling with LWP instructions (llwpcb/slwpcb) to load and store a LWP control block (LWPCB)
- LWPCB is in the processes memory region and specifies profiling details (selected events, ring buffer, current and max counter values)
- Interrupt support to notify if buffer runs full
- LWP hardware writes into LWPCB for sw interaction
- LWP hw state is saved to the LWP xsave area with each context switch

Lightweight Profiling (LWP), Events and Sample

- Events:
  - Instructions Retired event enabled
  - Branches Retired event enabled
  - DCache Miss event enabled
  - CPU Clocks Not Halted event enabled
  - CPU Reference Clocks Not Halted event enabled
- Sample:
  - event id
  - core id
  - Instuction address
  - time stamp
  - event specific data/flags

LWP kernel support

- Kernel support necessary
  – context switch support (xsave/xrestore)
  – Enable LWP instructions
- Patches posted by Hans Rosenfeld: https://lkml.org/lkml/2011/11/29/206
- Negative review feedback: https://lkml.org/lkml/2011/5/17/151
- Main reasons:
  – No interrupt support
  – Security implications of enabled LWP instructions
  – No integration into the perf_event_open() syscall
- perf syscall integration (prototype):
  – patches posted by Benjamin Block
  – https://lkml.org/lkml/2011/12/16/227
- In between xsave/xrestore code base changed, rework necessary for this too.
- So far there is no outlook when LWP will be upstream

Summary

- Core Performance Counters, since v2.6.39, all family 15h models since v3.5
- Northbridge Performance Counters, patches posted for v3.5, rework for v3.6
- IBS, kernel patches upstream in v3.5, small updates needed for full support
- LWP, patches posted but not accepted, so far no outlook when end how LWP will be supported

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2012 Advanced Micro Devices, Inc. All rights reserved.