



Performance monitoring and virtualization

Stéphane Eranian

HP Labs

Gelato Meeting, Apr 2007 – San Jose, CA

© 2005 Hewlett-Packard Development Company, L.P.
The information contained herein is subject to change without notice



Agenda

- PMU introduction
- PMU and virtualization, why bother?
- Usage models
- PMU virtualization
- System-wide monitoring in Xen
- KVM
- perfmon2 update

What is the PMU?

- Performance Monitoring Unit (PMU)
- Piece of CPU HW collecting micro-architectural events:
 - from pipeline, system bus, caches, TLB ...
- PMU is highly specific to a CPU implementation
 - Montecito PMU \neq McKinley PMU
 - Minimal set of architected features (4 counters, 2 events)
- PMU proven critical to solve performance problems
- PMU is impossible to emulate without prohibitive cost
 - must allow access to actual PMU in virtual environments
 - must expose exact underlying PMU model to guests

Itanium PMU state

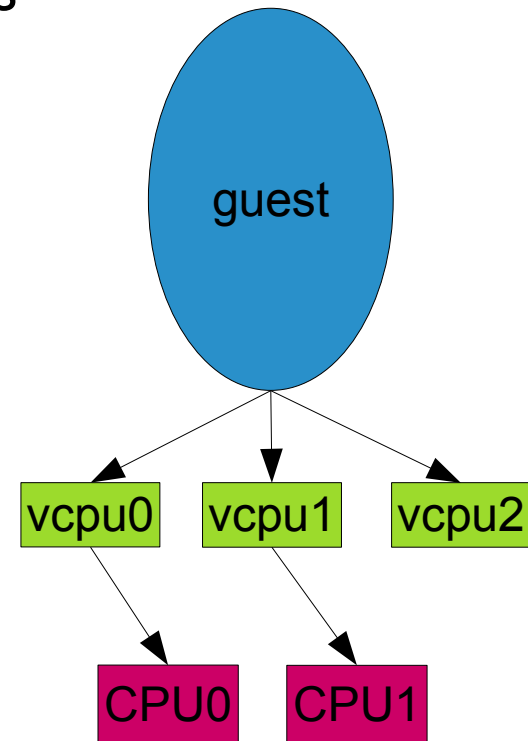
- Set of PMC and PMD registers (58 on Montecito)
 - fairly expensive to access (≈ 36 cycles)
- psr.pp/psr.up/psr.sp, dcr.pp
- cr.pmv
- optionally IBRO-7/DBRO-7

Usage models in virtual environments

- PMU usage is never for correctness but for performance
- Virtual environments to become mainstream quickly
execution incurs some overhead: not free!
must evaluate advantages vs. benefits?
- Ensure continuity of service
OS, applications using PMU must continue to work
Performance must be maintained: JVM with DPGO
requires per-guest PMU virtualization
- Assessment of machine performance
measure across hypervisor (VMM) and guest environments
requires system-wide PMU utilization

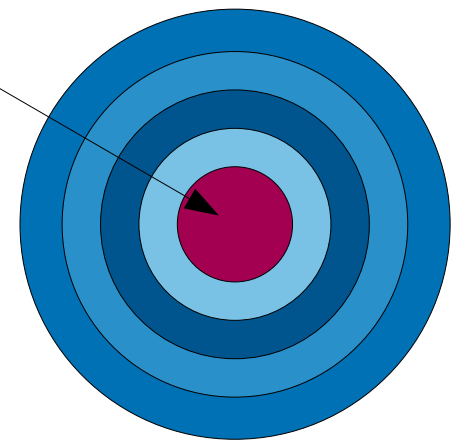
PMU virtualization

- PMU virtualization \neq perfmon virtualization
- Expose underlying PMU to guests OSes and apps
 - provide read/write access to PMU registers
 - propagate **raw** PMU interrupts to guest
 - multiplex PMU across guests
 - must work without guests modification
- PMU state and context switches:
 - PMU switch on virtual CPU (vcpu) switch
 - Cost mitigated using lazy save/restore



Architecture support for PMU virtualization

- “Normal” support (for para-virtualization):
 - at $plm \neq 0$:
 - do trap: read/write pmc, write pmd, .pp/.up/.sp
 - no trap: sum/rum psr.up, read pmd returns zero
- VT-i introduces a logical 5th privilege level with psr.vm
- VT-i support (with psr.vm=1):
 - at $plm = \text{any}$:
 - do trap: write pmc/pmd, read pmc
 - at $plm \neq 0$
 - no trap: read pmd returns 0
- PAL support: Virtual Processor Descriptor (VDP)
 - can disable traps on: write/read pmc, write pmd



Virtualization issues

- Allow monitoring while in VMM?
 - guest not supposed to know running on VMM
 - must stop/start on VMM entry/exit
- Privilege level exposure
 - native: 0=kernel 3=apps
 - Xen/para: 0=xen, 2=kernel, 3=apps
 - VT-i: 0=xen psr.vm=0, 0=kernel psr.vm=1, 3=apps psr.vm=1
- PMU exposes 4 levels in PMCx.plm
 - example: `pfmon -uk date (plm=0x9)`
 - native: user + kernel
 - para: vmm+user
 - VT-i : vmm+guest kernel+user
 - PMC writes need to be adjusted for plm

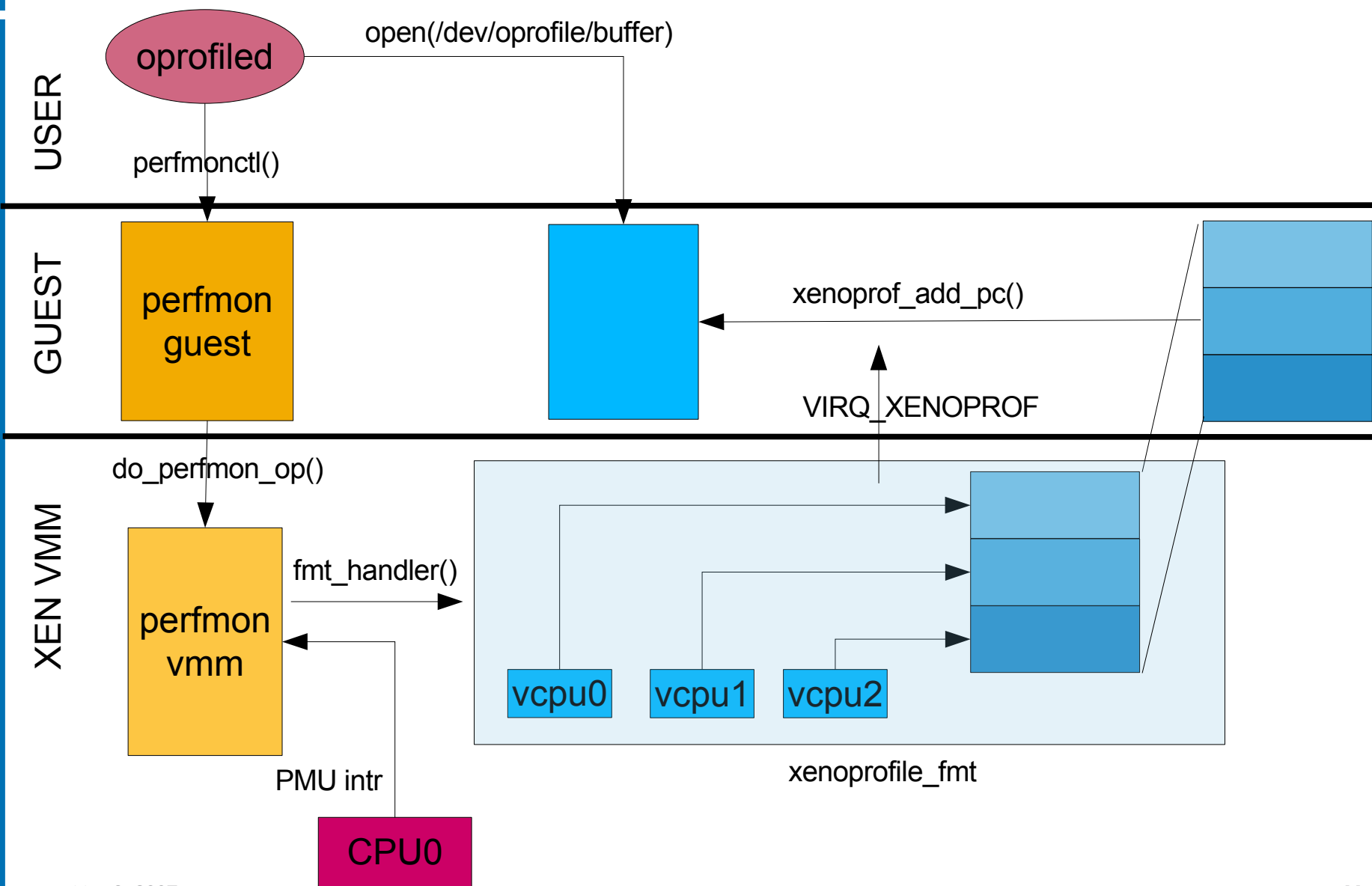
Virtualization and sampling

- Assume no monitoring while in VMM
 - only way to have unmodified guest support: perfmon
 - otherwise needs mechanism to pass VMM IIP to guest
- PMU interrupts trap to VMM
 - must forward back to guest
 - must be done immediately to ensure guest IIP accuracy

System-wide in Xen/ia64

- XenOProfile: extensions of Oprofile for Xen (HPLabs)
support for IA-64, i386, x86-64
- OProfile on Itanium:
uses perfmon to program PMU
gets PMU interrupt via custom sampling format
uses oprofiled and `/dev/oprofile` interface to extract data
- XenOProfile/ia64 (Isaku Yamahata, VALinux Japan)
must drop into VMM for PMU access
perfmon split: high level in guest, low-level in VMM
shared buffer between VMM and guests to pass samples

XenOProfile/ia64 architecture



Xen/ia64 VMM perfmon

- Limited to XenOProfile support
 - uses perfmon v2.0 interface and implementation
 - system-wide support only
 - no PMD read support
 - no changes for VT-i
- web site: <http://xenoprofile.sf.net>

PMU resource manager in VMM

- Generic, must work with para and VT-i
 - no perfmon abstraction/knowledge
 - support for unmodified guests (operate at instr level)
- Per-vcpu mode:
 - no monitoring when running in VMM
 - saves/restores PMU state on vcpu switches
 - immediately re-injects PMU interrupt in guest, no shared buffer
- System-wide mode:
 - monitoring while running in VMM
 - keeps PMU state across vcpu switches
 - uses shared buffer to communicate samples (IIP)
 - requires modified guest perfmon/oprofile subsystems
 - needs single point of control

PMU resource manager in VMM (2)

- How to allow concurrent vcpu/syswide sessions?
 - fail syswide if one vcpu exist?
 - probably okay
 - fail vcpu if syswide?
 - **hard to fail** an instruction
 - multiplex PMU HW under the cover?
 - autoscaling of results?
 - cooperation between users
 - give priority to PMU virtualization over syswide?
- Back to general PMU sharing problem!

Kernel Virtual Machine (KVM)

- Open-source project in mainline kernel since 2.6.20
 - x86-only at this point
 - very easy to setup and compile
- Uses Linux as VMM host
 - KVM is kernel module
 - uses modified `qemu` for device emulation, setup/tear-down
- Virtual machine runs as Linux process
- works only with VT(-x)/AMD-V
 - para-virtualized patches by Ingo Molnar exists

KVM and PMU

- System-wide mode: get it for free!
 - leverage host perfmon subsystem in syswide or per-thread
 - needs guest/KVM/qemu support for guest symbol correlations
- PMU virtualization using VT-i
 - trap PMC,psr accesses,PMD writes, propagate to perfmon
 - perfmon encapsulates guest PMU state for ctxsw
- PMU interrupts and IVT
 - comes in as external interrupt
 - guest IVT controlled by KVM (cr.iva)
- May need cooperation between host perfmon and KVM
 - catch and forward of interrupts

Perfmon status

- push to mainline to happen for 2.6.22
code fully reviewed by top-level maintainers
expected to stay in -mm tree for some time
- Perfmon v2.4 changes (over 2.2):
 - uses strings instead of UUID for sampling formats
 - `pfm_create_context()` returns file descriptor
 - system-wide: no more explicit start/stop in idle loop
 - IA-64: PFM_SETFL_EXCL_IDLE to explicitly stop
 - AMD64 (K8): NorthBridge event restriction per socket
 - MIPS: lots of updates from Phil Mucci
 - Cell processor port started (IBM), PPC updates (IBM)
 - Intel Core 2 Duo: full support incl. PEBS, NMI
 - early OProfile perfmon support on non-IA64 platforms

Conclusions

- Must ensure that PMU is not left out of VMM
- system-wide and virtual PMU modes are needed
- need a true PMU resource manager in Xen VMM
- KVM looks promising

<http://perfmon2.sf.net>



i n v e n t

Perfmon2: what is it?

- Kernel interface to access HW performance counters
- Very generic: not designed to support only one tool
- Supports per-thread and system-wide measurements
- Supports counting and sampling of HW events
- Efficient and flexible kernel level sampling buffer
- Support for events sets and automatic multiplexing
- Multi-architectures
- Selected to become the monitoring interface for Linux

Perfmon2 architecture summary



user level

