



Xenoprof overview & Networking Performance Analysis

J. Renato Santos

G. (John) Janakiraman

Yoshio Turner

Aravind Menon

HP Labs

Xen Summit

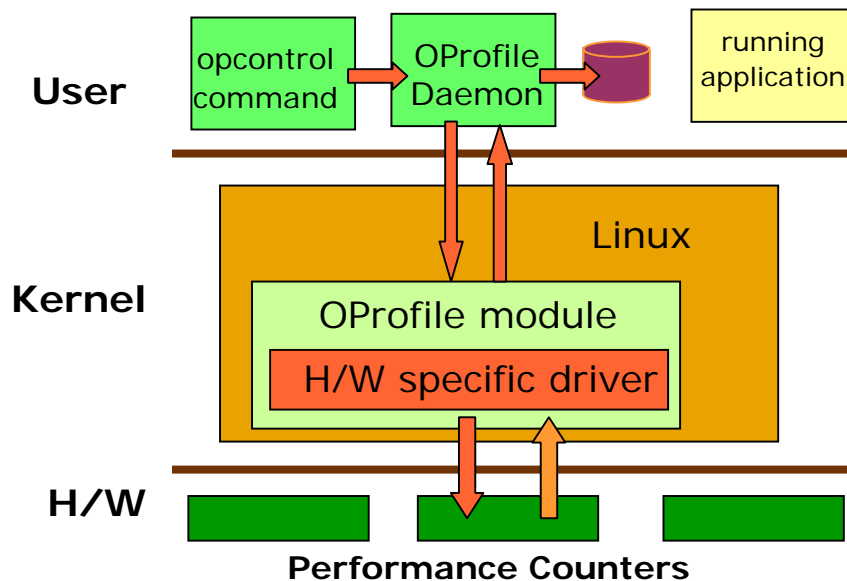
January 17-18, 2006



Background: OProfile Overview



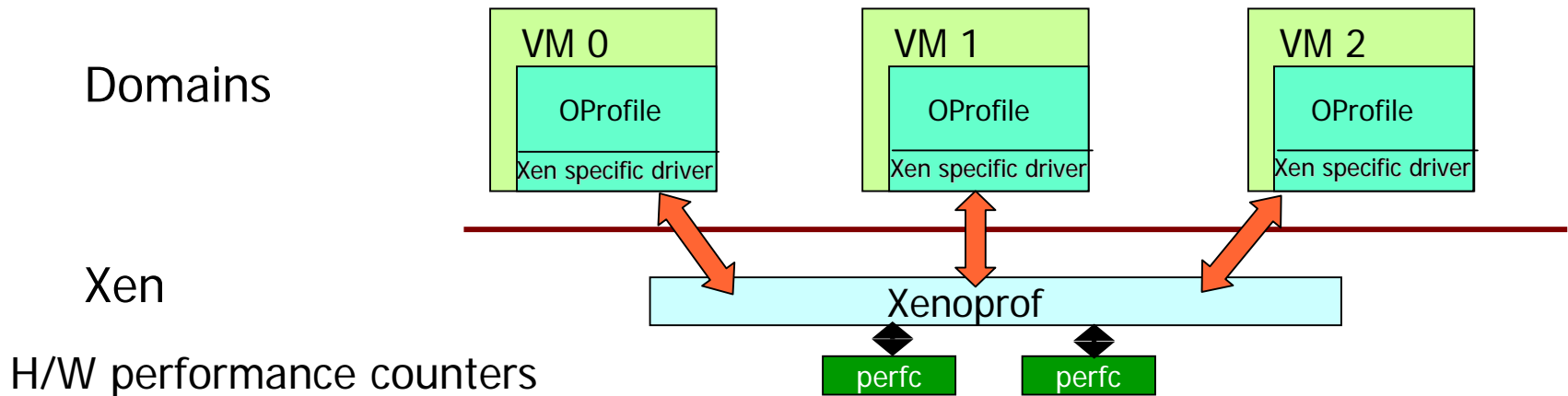
- Statistical profiling of applications on linux
 - Contribution of different routines in user/kernel space to execution cost
 - Profile various hardware events (clock cycles, instructions, cache misses, etc.)
 - Use NMI to sample code execution on perf. counter overflow



- Example output

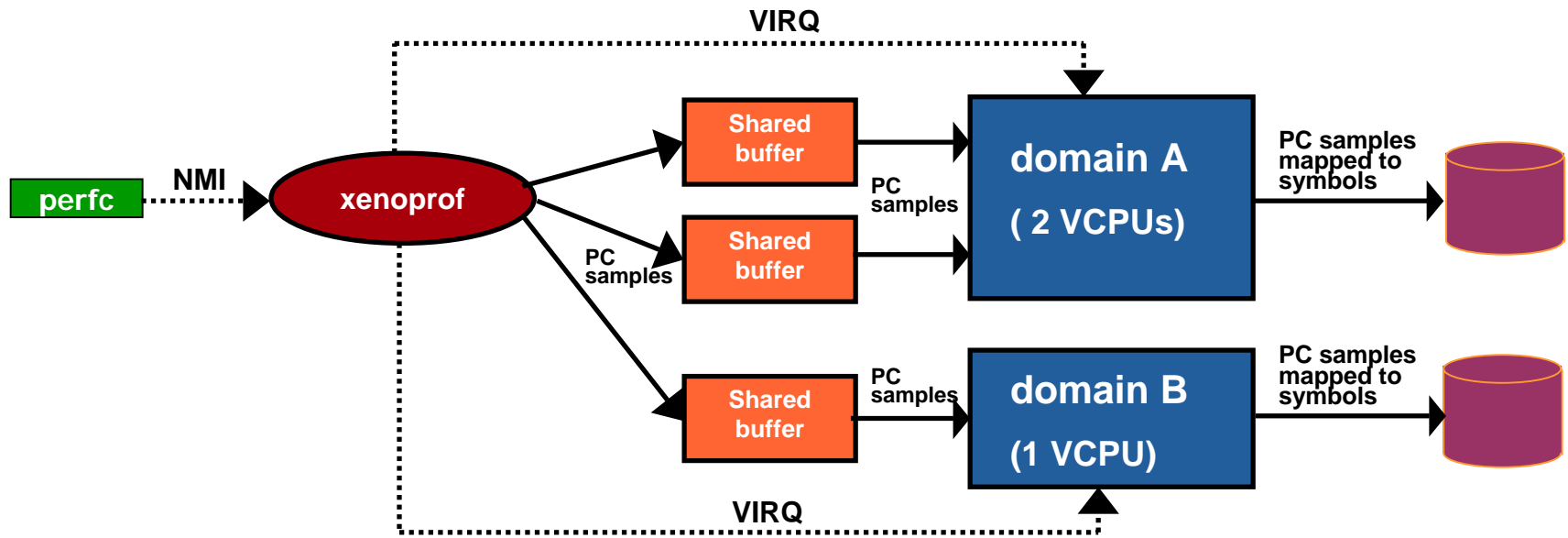
Function	%Instructions	Module
e1000_intr	13.32	e1000
tcp_v4_rcv	8.23	vmlinux
main	5.47	rcv22

Xenoprof: Enabling System Wide Profiling in Xen



- Xenoprof: Extends Xen with performance event interface
 - Map performance events to specific h/w perf counters
 - Hypercalls for setup, start, stop event sampling
 - System wide profiling (no perf. counter virtualization yet)
- OProfile: Extended with Xen specific driver

Xenoprof Architecture



- PC samples collected in Xen
 - Required to profile Xen code
- PC samples stored in shared buffer
- Domain notified via virtual interrupt
- Domain interprets samples obtained from shared buffer
 - Memory map of binary images is known only by the domain

Domain roles in a profiling session



- **Initiator** domain: (currently only dom0)
 - Coordinates profiling session
 - Defines profiling session setup (performance events, sampling period, profiled domain, roles of each domain, etc)
- **Active** domain:
 - A domain running OProfile
 - Interprets its own PC samples
- **Passive** domain: (not supported yet)
 - A domain not running OProfile (i.e. it doesn't interpret PC samples)
 - PC samples interpreted by other domain (initiator)
 - Coarse profiling for kernel/user level (no module/function information)
- **Unmonitored** domain
 - Domain ignored in profiling session (i.e. samples discarded)

Running OProfile in multiple domains



In coordinator domain (dom0):

```
> pcontrol --reset          # clean previous run samples
> opcontrol --start-daemon \ # setup session
  --active-domains=0,3,7 \
  --event=GLOBAL_POWER_EVENTS:1000000:1:1:1 \
  --vmlinuz=/boot/vmlinuz-syms-2.6.12.6-xen0 \
  --xen=/boot/xen-syms-3.0.0
```

```
> opcontrol --start        # start profiling
```

(run benchmark)

```
> opcontrol --stop        # stop profiling
```

```
> opcontrol --deinit      # stop OProfile daemon and
                          # and remove OProfile module
```

In other active domains (eg. 3,7) :

```
> pcontrol --reset          # clean previous run samples
> opcontrol --start \      # indicate domain is ready
  --vmlinuz=/boot/vmlinuz-syms-2.6.12.6-xenU \
  --xen=/boot/xen-syms-3.0.0
```

(run benchmark)

```
> opcontrol --deinit      # stop OProfile daemon and
                          # and remove OProfile module
```



- For OProfile report:

- Need to concatenate individual reports in each domain

- Currently supports X86 (P6 family and Pentium 4) only
- Xenoprof code include modification in 3 components
 - Xen, Kernel, OProfile user level tools
- Xen and Kernel components to be included in the Xen public repository, sometime soon ...
- OProfile maintainer (John Levon) has accepted user level modifications into OProfile. (will be available in the next public release of OProfile – no date defined yet).

- Port to additional CPU models
 - Currently supported : x86_32 (P6 family and Pentium4)
 - Need support for: AMD, x86_64 (future: IA64, PPC)
- Support for passive domains
- Add Performance Counter virtualization into Xen
 - Would enable individual domain profiling
- Enable HW counter access from domains
 - New domain privilege for accessing HW counter needed
 - Would enable use of additional tools
- Support for call-trace

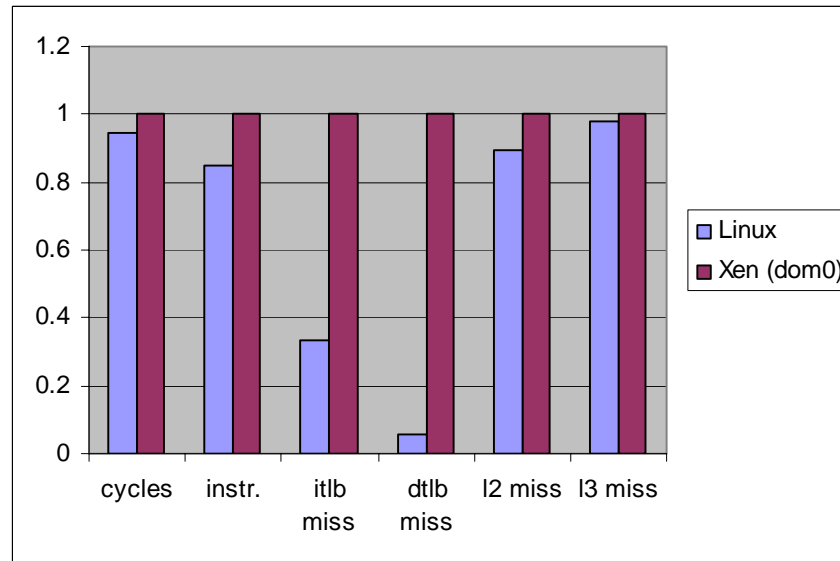
Networking Performance Analysis

- Experiment setup
 - Machine: HP Proliant DL580
 - 4-way P4 (Xeon),
 - 2.8 GHz,
 - 4 GB RAM (256MB/dom)
 - E1000 gigabit network card
 - benchmark
 - Single TCP connection (Receive side)
 - Large MTU-size packets
 - max throughput (930 Mb/s)

Dom0 has good performance for I/O



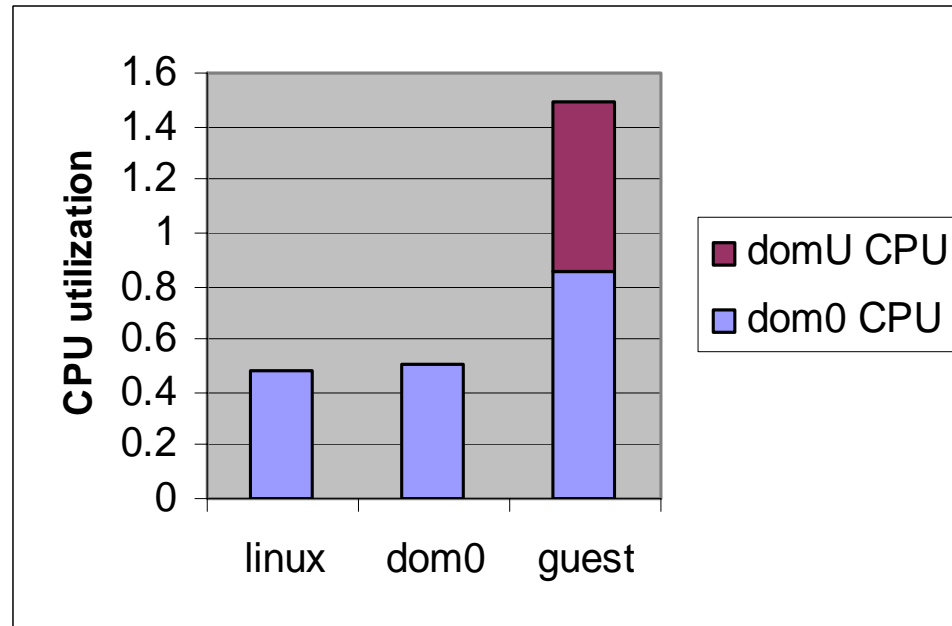
Total # samples in experiment (relative to Xen)



Kernel: 88.3%
Xen: 11.0%
User: 0.7%

- Dom0 has performance similar to Linux
 - ~5% additional overhead
 - Overhead mostly due to additional instructions in Xen
 - TLB misses significantly higher in Xen but negligible effect on performance

I/O in domU has high overhead



- Network I/O has significant CPU overhead in driver domain
- Need to optimize Xen I/O performance

code profile for dom0 (with application in domU)

Kernel (dom0)			Xen (in dom0 context)		
samples	%	class	samples	%	class
15020	11.8946	netback	11590	9.1785	arch/x86/mm.o
12670	10.0337	bridge	10967	8.6850	grant_table.o
9118	7.2207	net/core	5545	4.3912	find_domain_by_id
6958	5.5101	netfilter	4953	3.9225	arch/x86/x86_32
6003	4.7539	e1000	4418	3.4987	page_alloc.o
4646	3.6794	arch/xen/i386/kernel	4310	3.4132	io_apic.o
4209	3.3331	arch/xen/kernel	2314	1.8325	event_channel.o
3959	3.1352	ethernet	1710	1.3542	usercopy.o
3544	2.8071	mm	7817	6.1903	other
2840	2.2490	ip_tables	53624	42.4661	TOTAL
3367	2.6675	other			
72334	57.2843	TOTAL			

- Try to optimize high cost functions such as find_domain_by_id
- Bridge/net/ethernet code (~28%) responsible for approx half of kernel time
 - Should try to optimize bridge code (or use an alternative solution)
- memory management in Xen (mm/grant/page_alloc) also has high cost (21%)
 - Should try to optimize page flipping & page ownership exchange

- Enable direct I/O from domains running application
 - Each device is dedicated to one domain (not ideal)
 - Security issue: DMA can violate domain memory boundaries (until IOMMU is available)
- Virtualization support on devices
(sharing devices among domains with direct I/O)
 - Need adoption by device vendors
 - Industry is moving in this direction - PCI Express working on a standard extension for I/O virtualization



i n v e n t