



Xen Summit 2006

Xen/ia64 Session

Dan Magenheimer (djm@hp.com)
Hewlett-Packard Laboratories

© 2004 Hewlett-Packard Development Company, L.P.
The information contained herein is subject to change without notice





Xen summit

Xen/ia64 session agenda

- 2005 progress / status -- 10min (Dan Magenheimer)
- Xen/ia64 paravirtualization – 10 min (Dan)
- Xen/ia64 VT-i support – 10 min (Fred Yang)
- Working session – 60 min (led by Fred Yang)
 - physical-to-machine mapping
 - VHPT
 - interrupt handling
 - reboot/destroy
 - timer virtualization
 - other items as needed

Xen/ia64 2005 Progress / Status



Community growth

- January 2005: one person, prototype just released
- January 2006: xen-ia64-devel has 125 subscribers
- Multi-company, worldwide team
 - HP: paravirtualization, drivers, performance
 - Intel: VT-i, driver merging, tough bugs
 - Bull: SMP host support
 - Fujitsu: RHEL4, recipes, regression test suite
 - UNSW: block driver, xend/xm multiple domain support
 - VALinux: debugging support
 - CERN: early multi-domain support

Xen/ia64 2005 Progress / Status



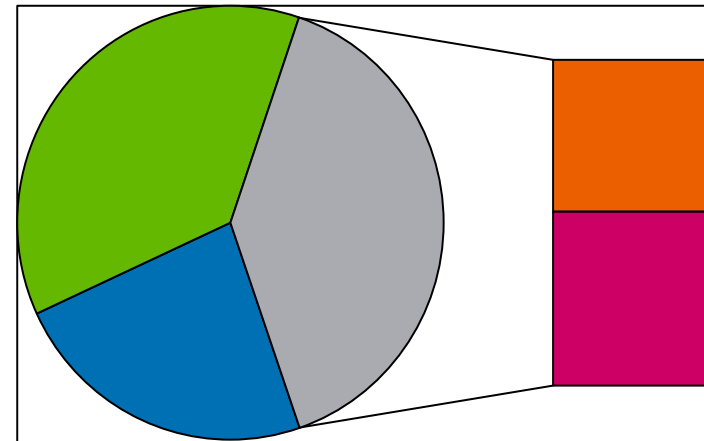
Multi-architecture Xen

- Driving Xen common/archdep boundaries
- Merging Xen/ia64 upstream
- Merging Xen/ia64 upstream
- Leverage from Linux/ia64
- RESULT:
 - Xen/ia64 included in Xen 3.0.0 release (Dec 05)!!!

Xen/ia64 open source code leverage

- Total code lines*: 83K
 - code (.c,.S): 50K
 - headers (.h): 33K
- High leverage
 - Shared with Xen/x86: 19K
 - Linux files 100% levg'd: 15K
 - Linux heavily levg'd: 17K
 - Null'ed Linux headers: 24 files
- Minor patches for linux files
 - context-diff is 1700 lines
 - lines added: 659 (<4%)
 - non-cpp lines added: 382
 - ~115 changes in 36 files
 - almost all are 1-2 line diffs

* all counts are "wc -l"



Xen/ia64 2005 Progress / Status



New functionality in 2005

- Paravirtualized Xenlinux
 - objective: low impact, high performance
- Multiple domain support
 - core hypervisor (scheduler, context switch, isolation)
 - xend + xm tools
 - Xen block driver / grant tables
- VT-i support
- SMP host support

Xen/ia64 2005 Progress / Status



Broader support

- Tested platforms:
 - HP rx26x0, rx16x0
 - Next generation Itanium
 - Intel Tiger4
 - Other “large” platforms
- Distros:
 - RHEL3 -> RHEL4
 - Debian
- Testing
 - regression suite running (Fujitsu)

Xen/ia64 2005 Progress / Status



2006 to-do list

- Track Xen and Linux releases
 - more Xen convergence or influence arch-neutral API changes
 - push Xenlinux/ia64 changes upstream?
- Xen network / balloon driver
 - delayed pending p2m vs arch-neutral driver discussions
- SMP guest support
- Save / restore / migration
- Driver domain support
- Broader set of platforms and distros (SLES? NUMA?)
 - eliminate physical memory constraints
 - major changes to core Xen needed?
- Other guest OS's
 - Windows (under VT-i)? HP-UX? ?
- Documentation improvements ☺

Questions?



i n v e n t

Xen summit

Xen/ia64 session agenda

- 2005 progress / status -- 10min (Dan Magenheimer)
- **Xen/ia64 paravirtualization – 10 min (Dan)**
- Xen/ia64 VT-i support – 10 min (Fred Yang)
- Working session – 60 min (led by Fred Yang)
 - physical-to-machine mapping
 - VHPT
 - interrupt handling
 - reboot/destroy
 - timer virtualization
 - other items as needed

Xen/ia64 Paravirtualization

Paravirtualization Objectives:

- Correctness (pre-VT)
- Performance
- Minimal impact to guest

Xen/ia64 Paravirtualization

Xen/ia64 guests run unprivileged (PL1) which means:

- All privilege-sensitive instructions must be either replaced with privileged instructions (“privified”) or by hypercalls
 - e.g. “cover”, “thash”, “ttag”
- All privileged instructions must be emulated or replaced with hypercalls

BUT...

- Privileged instructions pervade Linux → large change “footprint” → long-term maintenance headaches
- Emulation can be very slow; hypercalls too!
 - instruction fetch (from data cache!) and decoding is slow
 - getting to C is expensive
 - full emulation has many difficult corner cases (e.g. stacked registers?)

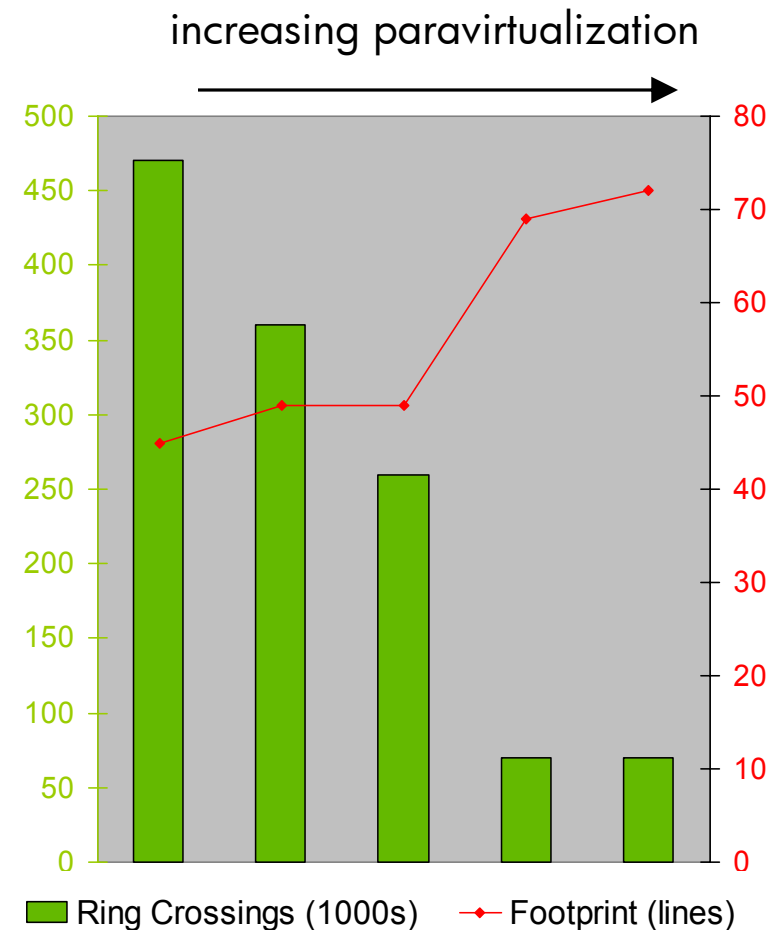
Xen/ia64 Paravirtualization

Solutions:

- Reduce footprint
 - Optimized paravirtualization
 - Transparent paravirtualization
- Innovative performance techniques
 - hyper-registers
 - hyper-privops
 - hyper-reflection

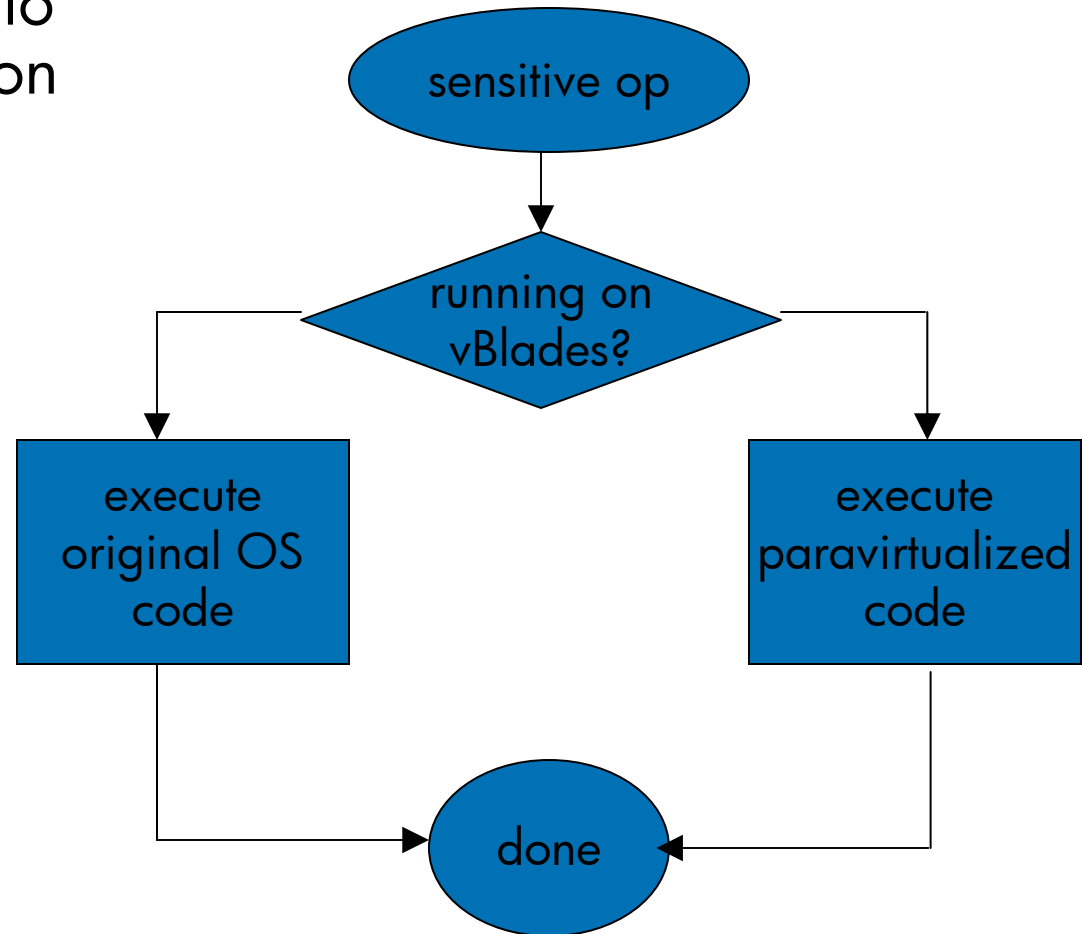
Optimized paravirtualization

- Supporting both virtualization and paravirtualization provides valuable flexibility
 - “80/20” tuning rule applies
 - most startup code can remain unchanged
 - minimizing changes is a good thing



Transparent paravirtualization

- Allows common binary to run both on VMM and on bare metal.
- Performance impact is negligible (< 0.1%)



Xen/ia64 paravirtualizing for performance

- Minimize ring crossings due to priv-reg accesses
 - Hyper-registers
- If VMM entry is necessary, ensure it is fast
 - Hyper-privops
- Ensure hardware interruption handling is efficient
 - Hyper-reflection

Xen/ia64 “hyper-registers”

- memory-mapped virtual privilege registers
- guaranteed “soft-pinned” for guest access
 - pointer available in ivt handlers in r31 (virtual bank0)
- often faster than accessing privileged registers!
- register def’ns not identical to physical regs, e.g:
 - some psr bits extracted to separate memory locations
 - cr.irr[0-3] -> “pending flag”
- semantics sometimes not identical either
 - bank0 and bank1 simultaneously accessible
 - pre-cover and post-cover virtual cr.ifs both available
 - writes may require “sync” (i.e. hypercall to Xen)

Xen-ia64 hyper-registers

Example: Turning off/on interrupts

Itanium disabling interrupts

```
rsm psr.i
```

Itanium enabling interrupts

```
ssm psr.i
```

Xen disabling interrupts

```
movl r2=virtual_psr_i  
st4 [r2]=r0
```

Xen enabling interrupts

```
movl r2=virtual_psr_i  
movl r3=virtual_pend  
movl r1=1 // enable  
st4 [r2]=r1  
ld4 r1=[r3]  
//pending?  
cmp.ne p6,p0=r1,r0  
(p6) hyper_ssm_psr_i
```

Xen/ia64 “hyper-privops”

Problem: Need a *fast* way to do simple hypercalls

Issues:

- immediate values for `break` may be owned by guest
- `epc` instruction creates lots of restrictions
 - e.g. can't take interrupts on non-privileged stack
 - see `linux/Documentation/ia64/fsys.txt`

Solution: hyper-privops

- `break` instructions with “virtual `psr.ic`” off
- special calling convention for args (r8-r11)

Xen/ia64 hyperprivops

Example: Setting a region register

xen_set_rr:

```
movl    r11=virtual_psr_ic
mov     r8=r32
mov     r9=r33;;
ld8     r10=[r11] // preserve virt psr.ic+psr.i
st8     [r11]=r0;; // disable virt psr.ic+psr.i
break   HYPER_SET_RR;;
st8     [r11]=r10;; // restore virt psr.ic+i
br.ret.sptk.many rpi;
```

Result: Many high-frequency hypercalls can be handled in Xen without leaving “bank 0” mode or turning on interrupts

Xen/ia64 hyper-reflection

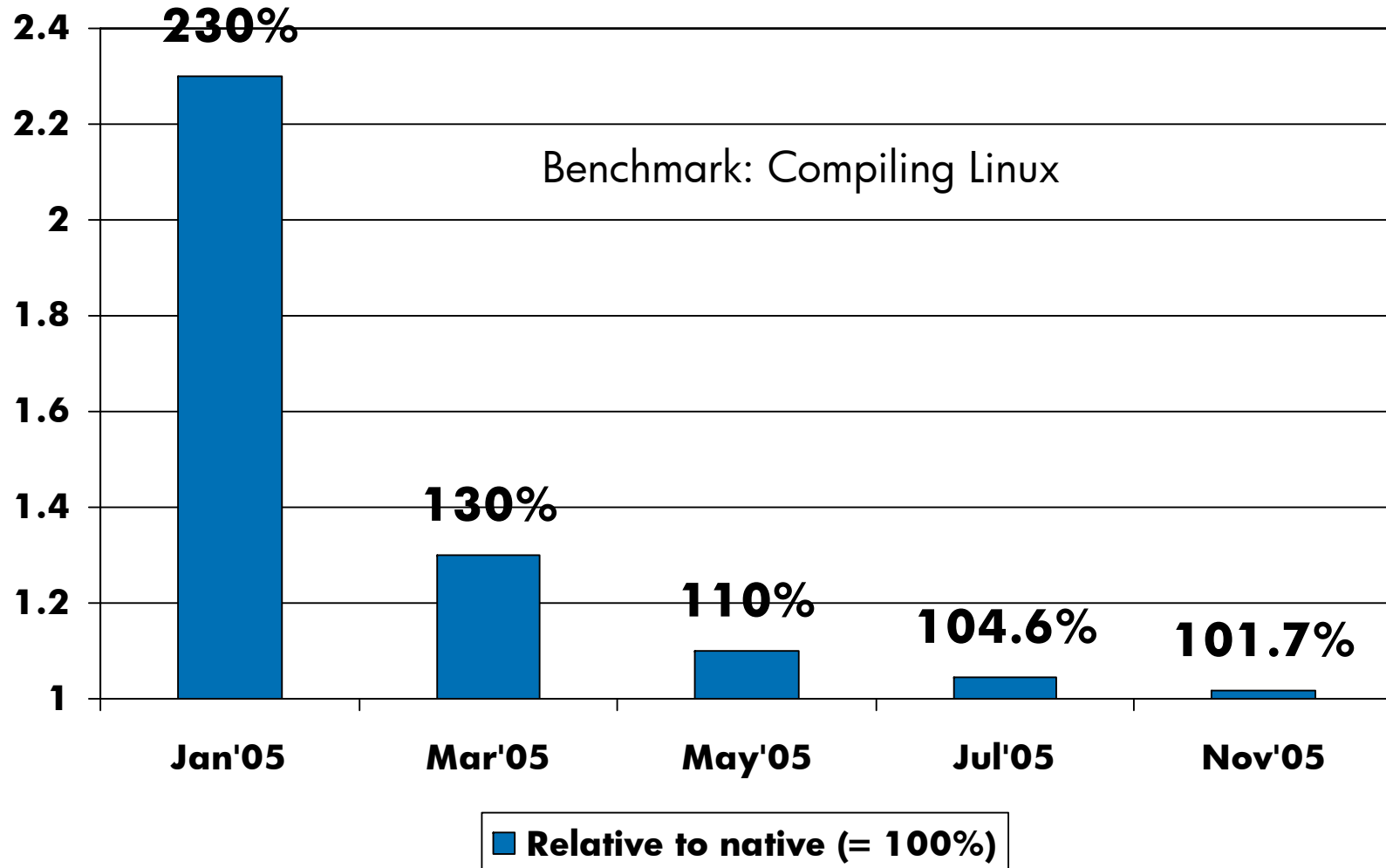
Many architecturally-defined “interruptions” must be handled by the guest, but:

- Xen must often “translate” interruption parameters

Techniques:

- Keep nearly all data structures in “pinned” memory
- Handle high-frequency cases in bank0 without going to C
- Assume unpinned data structures are in cache/TLB, but prepare fast recovery if they are not

Xenlinux/ia64 2005 Paravirtualized Performance



Questions?



i n v e n t

Xen summit

Xen/ia64 session agenda

- 2005 progress / status -- 10min (Dan Magenheimer)
- Xen/ia64 paravirtualization – 10 min (Dan)
- **Xen/ia64 VT-i support – 10 min (Fred Yang)**
- Working session – 60 min (led by Fred Yang)
 - physical-to-machine mapping
 - VHPT
 - interrupt handling
 - reboot/destroy
 - timer virtualization
 - other items as needed