

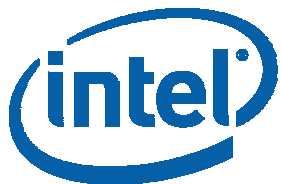


Xen/ia64 Next Steps Discussion Xen Summit

January 17, 2006

Kevin Tian

Fred Yang



Topics

- **Physical memory for Dom0**
- **Virtual interrupt controller**
- **VTLB/VHPT SMP support**
- **Reboot/Destroy**
- **Hypercalls**
- **Timer virtualization**
- **Things to improve**

Physical memory in Dom0

- **Currently P=M physical in Domain0**
 - Fewer changes to get XenLinux, with
 - Issues on reuse common Xen drivers
 - VBD needs customization to work & VNIF has page flipping issue
 - Balloon driver issue
 - How to handle Driver Domain?
 - Proposing Xen driver change or Driver API support
- **P2M alternative**
 - Proven XenLinux/x86 design with more Linux modification
- **VP (Virtual Physical) alternative**
 - P2M handled within Xen to keep minimal XenLinux changes
 - Need special DMA handling or Driver API support

Physical Memory (cont.)

Thread extracted from Xen-devel essay discussion

- **P=M**

- + Fewer changes in Xenlinux; easier to push upstream
- + Making Xen more flexible is a good thing
- ? May provide better foundation for future features (oversubscr, NUMA)
- More restrictions on driver domains
- More hacks required for some Xen drivers, or
- More work to abstract and define a portable driver architecture abstract

- **P2M**

- + Fewer differences in Xen drivers between Xen/x86 and Xen/ia64
- + Easier to implement remaining Xen drivers for Xen/ia64
- Major changes may require months for Xen/ia64 to regain stability
- Many more changes to Xenlinux/ia64; more difficulty pushing upstream
- No attempt to make Xen more resilient for future architectures

Memory enhancement

- **Page reference count**

- Dom0 Get/Put_page are no-op
 - Dom0 can access any machine frames without Xen aware
- Possible stability issue and affect domain destroy

- **Writable page table**

- Is it a good thing to go for Xen/ia64?
 - Bunch of Xenlinux/x86 changes come under this design
 - However Domain managed page table is not walked by IA64 processor
 - We are only maintaining virtual TLB within Xen/IA64

Virtual interrupt controller

- **Current physical interrupt controller is owned by Dom0**
 - Events are bound to a hard-code interrupt vector,
 - Event handler is interrupt injected into linux's interrupt sub-system
 - Pros:
 - No change to xenlinux interrupt sub-system
 - Cons:
 - Driver domain support
 - Multi-guest support in a domain
 - Difficult to have Xen own its devices, like serial
- **Virtual interrupt controller proposal for Xen to own physical controller (IOSAPIC)**
 - Necessary to support driver domains
 - IPI for SMP guest support can be bound to event
 - X86 implementation has already there
 - Consider Callback/Failsafe for XenLinux
 - After converted physical interrupt to event

VTLB/VHPT SMP Support

- **No uniformed VTLB/VHPT between para-virtualized domains and DomainVTI**
 - Para-Domains take per-LP VHPT design with 1 VTLB
 - DomainVTI takes per-VP design with Hash VTLB

- **How to address collision chain?**

- **How to address SMP guest?**

Reboot/destroy

- **No reboot support**
 - Reset para-driver state in xenstore
 - Setup context for reboot
 - Missing hypercalls
- **Domain destroy is not ready**
 - Need Page reference count

Hypercalls

- **Catch up with common hypercalls as much as possible for performance**
 - HYPERVISOR_sched_op (cpu hotplug/idle/reboot)
 - HYPERVISOR_multicall (netback/netfront/performance)
 - HYPERVISOR_physdev_op (virtual interrupt controller)
 - HYPERVISOR_vcpu_op (Guest SMP support)
 - HYPERVISOR_suspend (Reboot)
 - HYPERVISOR_set_timer_op (tick-less in idle loop)
- **Memory related hypercalls can be deferred per writable page table decision**

Timer virtualization

- **Currently virtual timer interrupt is raised periodically**
 - Do we need to change it to tick-less by adding timer related hypercalls?
 - Is it worthy of effort to modify Domain time sub-system?