



Virtualization on the SGI Altix

**Jes Sorensen, SGI
Gelato ICE, October 1-2, 2007, Singapore**

Agenda

- Virtualization typology
- Hypervisor strategies
- Current situation on Altix
- What the future will bring
- Questions (and possibly answers)

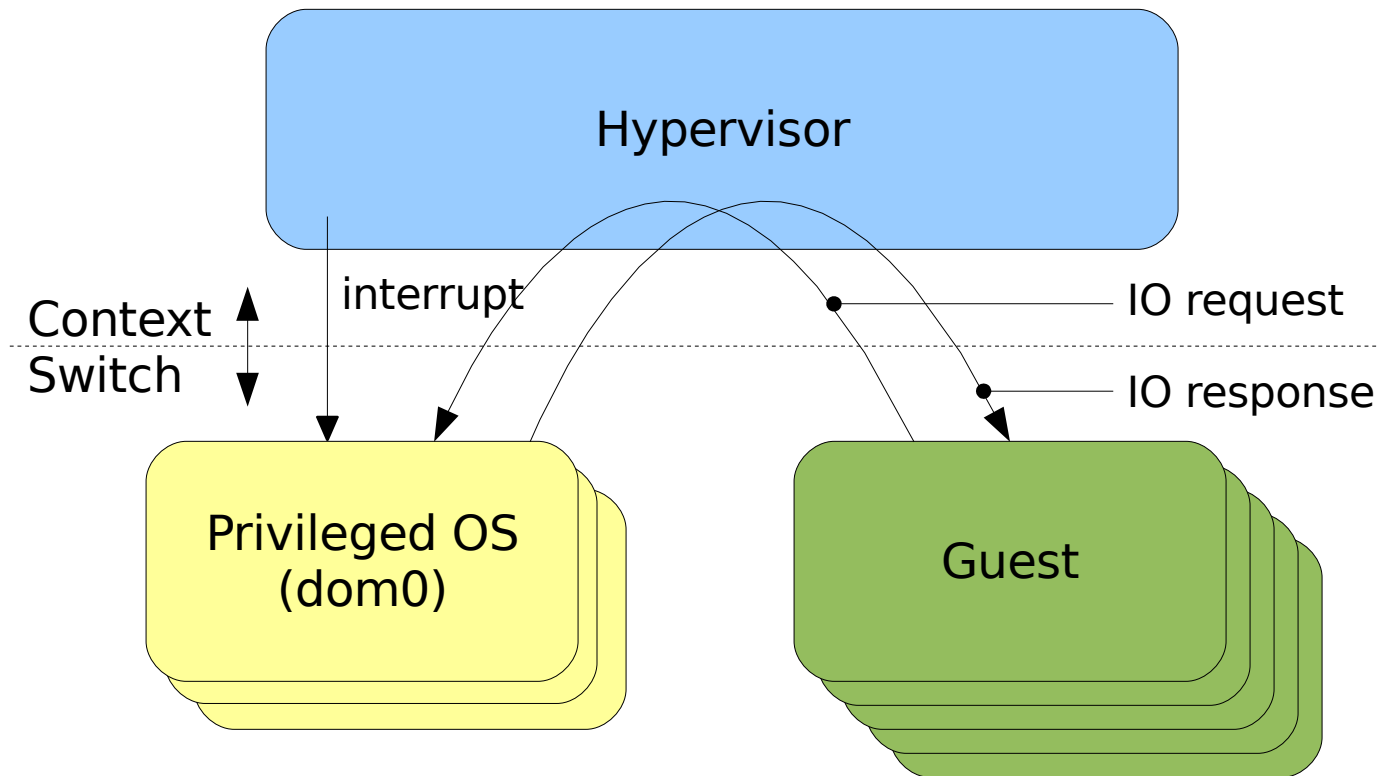
Virtualization Typology

- Para-virtualization
 - Guest OS aware it is run under hypervisor
 - Hypercalls to call into hypervisor (device drivers, page table updates etc)
 - Some instructions are still emulated
- Full-virtualization
 - Guest OS is unaware it is running under hypervisor
 - All emulated / trapped (device drivers, page tables etc).
- Full-virtualization with para support (combined)
 - Device drivers use hypercalls for performance speedups (VMWare style)
- Containers
 - Not virtualization – provides multiple instances of same OS

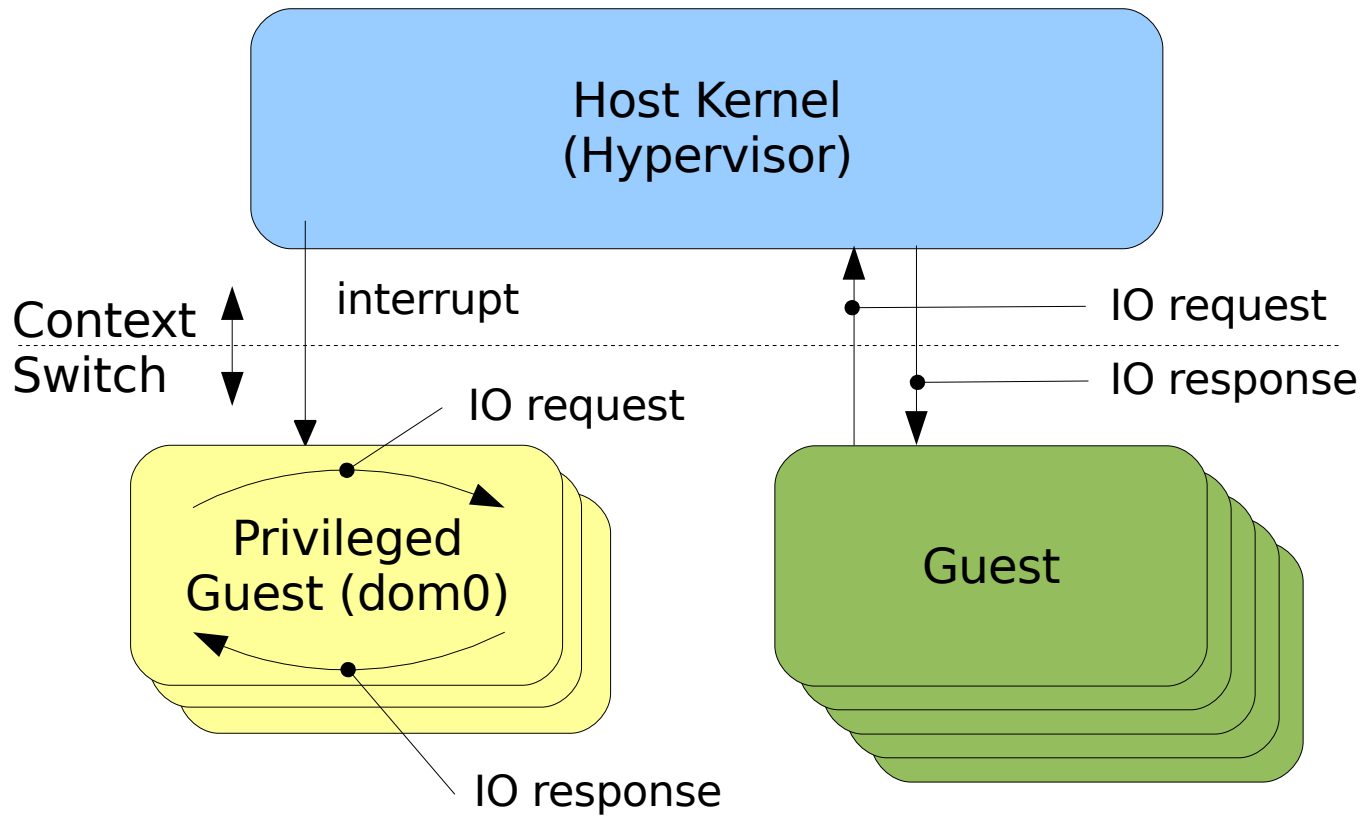
Different hypervisor models

- Microkernel based
 - Hypervisor runs on the bare hardware
 - Xen style
 - Hypervisor provides basic device drivers
 - Use privileged guest to perform heavy lifting (Xen's dom0)
 - Alternative hypervisor provides full I/O support
- OS as hypervisor
 - The Operating System operates as hypervisor
 - Virtual machines runs as tasks – normal scheduler
 - I/O performed by host operating system
 - Privileged guests with direct access to specific hardware also possible (dom0 style)

I/O in Micro-kernel hypervisor



I/O with kernel based Hypervisor



Hypervisor model comparison

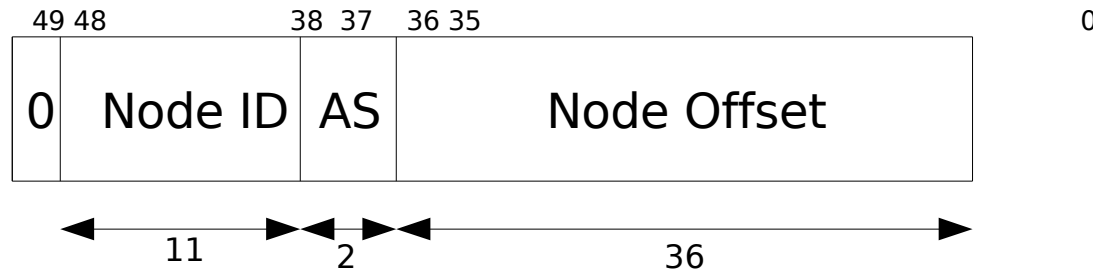
- Kernel based
 - No context switch for host OS' I/O operations
 - Two context switches per guest I/O
 - Context switch for interrupt delivery with privileged guest I/O
 - Microkernel based
 - Context switches for all I/O operations
 - Four context switches per guest I/O
 - Context switch for interrupt delivery for all I/O (privileged and regular)
- One context switch counted as switch from guest to hypervisor or from hypervisor to guest
 - Context switch and interrupt overhead for privileged guests comparable between the two models

What makes an Altix different?

- Large # of nodes, CPUs (2048 cores), I/O slots
- One Shub chip per node
 - Connects to CPUs, memory, I/O, and NUMALink
- No regular PC style basic I/O:
 - VGA
 - Serial 16c550
- No flat system bus
 - system topology needed to access any PCI device, including VGA
- System console accessible via SAL call
- IPI and TLB flush handled via Shub register write

More Altix Characteristics

- ITC (TSC on x86) not synchronized across nodes!
- Highly sparse memory layout, Shub addressing:



- AS: Address Space, 0b11 == Cacheable (normal)
- Actual physical memory map:
 - Node 0: 0x003000000000-0x004000000000
 - Node 1: 0x00b000000000-0x00c000000000
 - Node 2: 0x013000000000-0x014000000000
 - Node 3: 0x01b000000000-0x01c000000000
 - Node 4:

Current Altix Virtualization

- Xen has been booted in the lab on Altix 3700 series systems
 - 32 nodes (256GB of memory)
 - 64 CPUs
 - dom0 + a couple of domU's booted
- Work to do before suitable for users:
 - Shub2 (Altix 4700) support
 - NUMA memory placement for dom0 + domU
 - CPU affinity: Bind vCPUs to physical CPUs)
 - Interrupt affinity: Requires vCPU affinity
 - Larger memory support
 - Review of all hw workarounds implemented in Linux kernel, including I/O support
 - Support for drifting ITC

Future & ongoing Virtualization work

- Support efforts porting KVM to ia64
- Port Iguest to ia64
 - Implement hypercall interface library (for KVM/Xen/Iguest)
- Investigate open NUMA virtualization issues:
 - Memory handling/affinity
 - CPU and interrupt affinity
 - How to handle guest migration vs affinity?
 - Scalability issues
 - Most efficient way(s) to present virtual machine to guest
 - What options to provide for guest virtual machine layout
 - NUMA vs simple SMP for legacy OS support

Questions?