Lecture 7

Xen and the Art of Virtualization

Paul Braham, Boris Dragovic, Keir Fraser et al.

Advanced Operating Systems

16 November, 2011

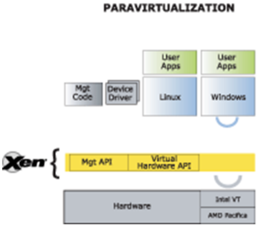
Contents

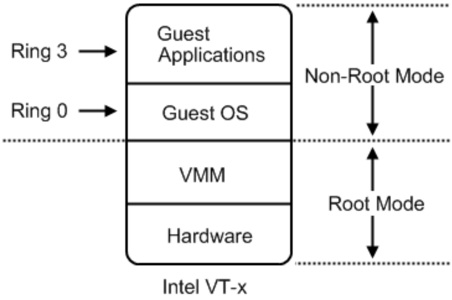
- creating multiple instances of virtual machines on a single
- physical server
- each virtual machine runs a separate operating system
- host system - runs on the server
- guest system - runs in the VMs

- full virtualization: the virtual machine fully emulates the hardware such that the guest OS can be run unmodified on top of it (Parallels, VMware Workstation, VMware Server, Virtual PC, QEMU)
- paravirtualization: the guest operating system needs to be modified, but the user applications need not

- Xen, VMware ESX Server
- Benefits
  - x86 was not designed with virtualization support
  - some privileged instructions do not generate a trap
  - VMware ESX server rewrites guest OS code to insert the traps

- paravirtualization – offer a VM interface similar to the hardware but not identical
- The ABI needs to remain unchanged
- open-source
- x86, x86-64, PPC
- The host system is a modified Linux or NetBSD
- Latest versions use Intel VT-x și AMD Pacifica – able to run an unmodified guest OS


PARAVIRTUALIZATION
User Apps
User Apps
Mgt Code
Device Driver
Linux
Windows
Xen
Mgt API
Virtual Hardware API
Hardware
Intel VT
AMD Pacifica


Ring 3
Ring 0
Guest Applications
Guest OS
VMM
Hardware
Non-Root Mode
Root Mode
Intel VT-x

- Hypervisor or VMM (Virtual Machine Monitor): the host operating system (Linux or NetBSD) with Xen support
  - operates at a privilege level higher than supervisor
- Domain: Xen VM
- Guest OS: instance of a guest operating system running in a VM

- Memory management
- CPU
- Devices

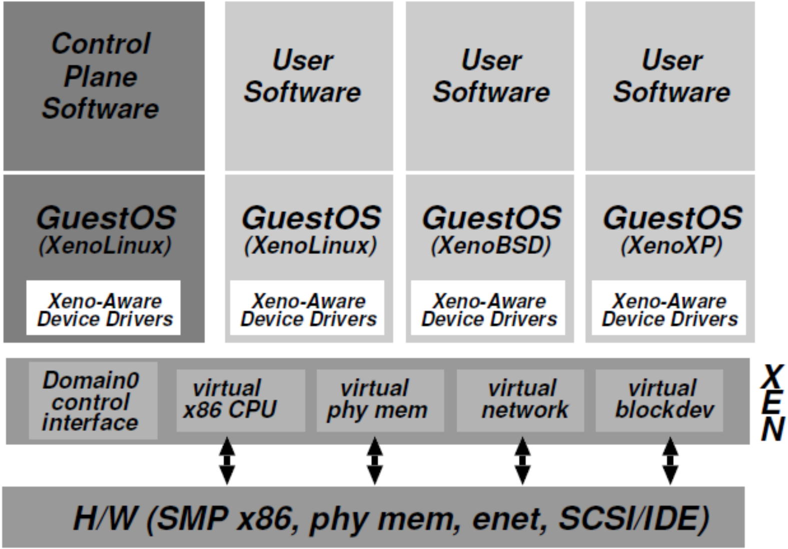
- No software managed TLB in x86
  - TLB misses are served by the processor by walking the page tables
- No tagged TLB (like on Alpha, MIPS, Sparc)
  - the tag can identify the address space: guest or host
  - any execution transfer requires a TLB flush
- decisions
  - guest OSes are responsible with allocating and managing the page tables
  - Xen exists in a 64M address space at the top of every address space, thus avoiding TLB flushes when entering and leaving the hypervisor

- OS -> hypervisor -> hardware
- The OS is no longer the entity with the highest priority
  - the OS runs at a lower priority level than the hypervisor
- X86 has 4 rings, 4 levels of privilege
  - apps run in ring 3
  - the OS runs in ring 1
  - hypervisor in ring 0
- The OS cannot run privileged instructions, they are paravirtualized and run in Xen

- Exceptions: memory faults, software traps, etc
- They are virtualized by creating handler tables for all traps inside Xen
- When an exception occurs while executing outside ring 0, Xen's handler creates a copy of the stack frame on the guest OS and returns execution to its handler
- frequent exceptions: system calls and page faults
  - for system calls - every guest OS installs a direct handler
  - cannot be done for page faults - need to read CR2 for the fault address - need to be handled through Xen

- Xen exposes a set of clean device abstractions
- information passing between domains is done through shared memory, asynchronous buffer-descriptor rings

- The hypervisor exposes basic operations
- Complex decisions are taken in management programs running in the guest OS
- Domain0 is created at startup:
  - has access to control operations
  - creating and terminating other domains
  - physical memory allocation
  - scheduling parameters
  - access to device I/O
  - create virtual interfaces (VIF) and virtual block devices (VBD)

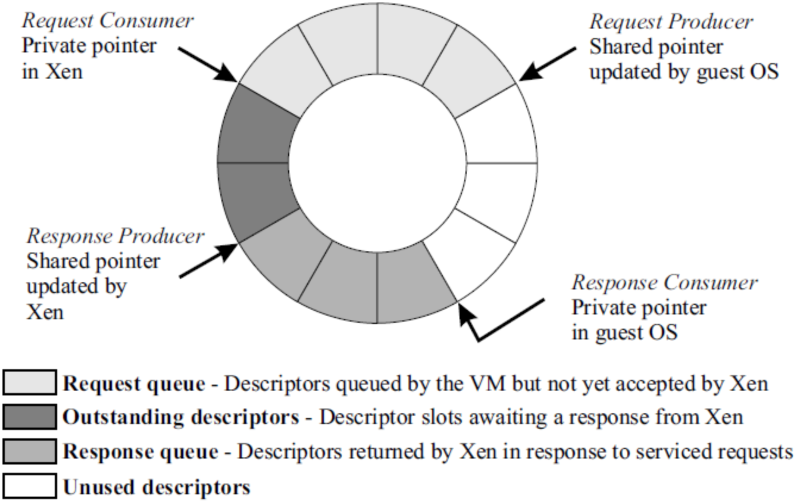

Control Plane Software
User Software
User Software
User Software
GuestOS (XenoLinux)
Xeno-Aware Device Drivers
GuestOS (XenoLinux)
Xeno-Aware Device Drivers
GuestOS (XenoBSD)
Xeno-Aware Device Drivers
GuestOS (XenoXP)
Xeno-Aware Device Drivers
Domain0 control interface
virtual x86 CPU
virtual phy mem
virtual network
virtual blockdev
XEN
H/W (SMP x86, phy mem, enet, SCSI/IDE)

- Hypercalls and events
- hypercall
  - trap software in the hypervisor
  - equivalent to a syscall
  - example: request a set of page-table updates
- communication from Xen to a domain is done through an asynchronous event mechanism – equivalent to interrupts
  - packet received from network
  - disk operation completed

Data transfer

- OS -> hypervisor -> device I/O
- I/O buffer rings
- a buffer ring contains I/O descriptors
- data is not in the buffer rings, instead is allocated by the guest OS and referenced by the descriptor
- producer-consumer pointers:
  - request producer (guest OS) - request consumer (Xen)
  - response producer (Xen) - response producer (guest OS)

Asynchronous I/O rings



- CPU scheduling
  - Borrowed Virtual Time (BVT)
    - chosen because it has a special mechanism for low latency dispatch of a domain
- Time
  - Xen offers guest OSes:
    - real time – nanoseconds since system boot
    - virtual time – advances when the domain is executing
    - wall-clock – offset to be added to real time

- address translation virtualization
  - VMWare provides each guest OS a separate virtual page table, not visible to the MMU
    - the hypervisors traps the access to this table and propagates the changes back and forth
    - overhead
  - Xen allows direct use of MMU accessible page tables
    - pages are marked read-only
    - updates are sent to Xen using a hypercall
    - for validation, physical frames have associated a usage counter and a type (exclusive values): PD, PT, LDT, GDT, RW
    - a physical frame cannot contain a page table and be also read write

- initial reservation specified at the time of domain's creation
- maximum memory can be specified
- use a "balloon driver" to pass memory pages back and forth

- Xen offers a VFR (Virtual Firewall Router)
- Each domain has one ore more VIFs in this router
- Each interface has associated two buffer rings - transmit and receive
- Each direction has rules associated for each direction: $<$pattern$>$, $<$action$>$
- Domain0 adds and deletes rules
  - prevents IP spoofing
  - demultiplex based on destination, port
  - firewall

- Transmit
  - Guest OS enqueues a packet on the transmit ring
  - Xen copies the header and runs the transmit rules
  - round-robin packet scheduler
  - scatter-gather DMA for payload
- Receive
  - Xen checks receive rules
  - destination VIF is determined
  - exchange the packet buffer with a physical frame in the receive ring
  - if no frame is available, packet is dropped

- Only domain0 has unlimited access to physical disks
- The other domains access the disk through VBDs
- VBDs are created by the management software running in domain0
- A VBD is accessed through the same I/O ring mechanism
- The OS disk scheduling system reorders requests prior to enqueuing them
- Xen also supports another pass of scheduling/reordering
- VBDs are serviced round-robin

Evaluation

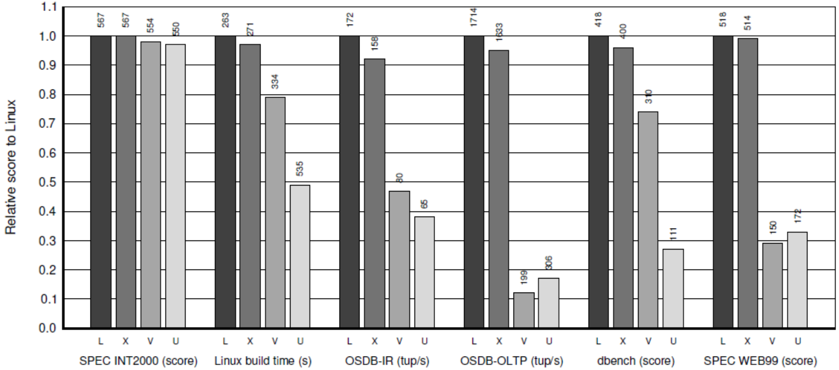


Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

Evaluation - bandwidth

	TCP MTU 1500		TCP MTU 500	
	TX	RX	TX	RX
Linux	897	897	602	544
Xen	897 (-0%)	897 (-0%)	516 (-14%)	467 (-14%)
VMW	291 (-68%)	615 (-31%)	101 (-83%)	137 (-75%)
UML	165 (-82%)	203 (-77%)	61.1(-90%)	91.4(-83%)

Table 6: `ttcp`: Bandwidth in Mb/s

- virtualization
- paravirtualization
- host and guest OS
- domain
- hypercall
- virtual interfaces
- virtual block devices
- privilege level

- http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf
- http://www.xen.org/
- http://en.wikipedia.org/wiki/Xen
- http://en.wikipedia.org/wiki/Virtualization

?