Xen and the Art of Virtualization

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield

Presented by: Karan Sharma
Full Virtualization vs Paravirtualization

- **Full Virtualization**
  - Guest OS are run unmodified on the Hypervisor
  - Guest OS are not aware that they are running on simulated hardware – Resulting in poor performance
  - Executing privileged instructions in non-privileged mode fails silently rather than causing a trap
  - Example – VMware, Disco

- **Para-Virtualization**
  - Guest OS are modified to run on the Hypervisor
  - Unlike Full Virtualization – sensitive instructions are NOT trapped, rather Hypercalls are made to the Hypervisor
  - Example - Xen
Full Virtualization vs Paravirtualization

Tanenbaum (2008)
Control Management

- **DomainO**
  
  Created at boot time
  
  responsible for hosting the application-level management software
  
  Can create and destroy other domains
  
  Controls other domain's associated scheduling parameters, physical memory allocations and the access they are given to the machine's physical disks and network devices
  
  supports the creation and deletion of virtual network interface (VIFs) and block devices (VBDs)
CPU Virtualization

- X86 – Four privilege levels
  - ring 0 is most privileged
  - ring 3 is least privileged
  - OS code executes in ring 0, and application code in ring 3, ring 1 and 2 are unused
  - Privileged instructions can only be executed in ring 0
- Xen approach – OS code executes in ring 1, hypervisor in ring 0
  - Hence, Guest OS is modified anywhere it uses Privileged Instructions and instead makes Hypercalls

Borrowed Virtual Time Algorithm
CPU Virtualization

- Exceptions are propagated to the guest from xen via event channels.
- Exceptions from system calls call directly from application into the guest OS.
  - Guest OS registers a 'fast' exception handler.
  - xen validates the address as part of the guest address.

Memory Virtualization

- Most Difficult part of Xen
- x86 does not have software managed TLB
  Software managed TLBs can be efficiently virtualized easily
  A tagged TLB – No TLB flush on a context switch

- Hardware Managed TLB
  Hardware managed TLB misses are serviced by processor in hardware
  No Tagged TLB – Complete TLB Flush

Solution:

- Guest OSes allocate and manage hardware page table with minimal involvement from xen
- Xen exists at top 64 MB of every virtual address space – saves TLB flush when entering and leaving the hypervisor

Virtualization of segmentation is similar
Virtual Address Translation

- Full Virtualization
  Shadow page tables – Two levels of indirection - increases overhead

- Paravirtualization
  Guest OS has direct access to hardware – reducing overhead and complexity
  Guest OSes have read-only access for page tables

**For write access:**
Goal: Isolation
Xen must make sure that Domains do NOT access each other’s memory
Each Domain must access its own memory

Solution: Page table updates are passed to xen via hypercalls – xen validates all requests, hence, ensuring safety and isolation
Physical Memory Allocation

- Balloon Driver concept
  Each Domain (OS) can adjust its allocated memory up to its maximum allowable limit
  However, there are provisions for requesting additional memory
- Xen does not guarantee contiguous allocation of physical memory – allocation at hardware level may be sparse
  Domains think that they have contiguous allocation of physical memory – There is a two-way mapping to accomplish this
Device I/O

- I/O Rings – Reduce overhead
Networking

- Virtual Firewall Router
- VIF (Network Interfaces)
- Disks
- Virtual Block Devices
Control Transfer

- Domains interact with Xen through hypercall
  - Synchronous software trap
- Xen respond to domain using event mechanisms
  - Asynchronous event mechanism

Extracted from http://www.cs.pdx.edu/~walpole/class/cs533/winter2010/slides/7b.ppt
Relative Performance

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

Simultaneous SPEC WEB99 Instances on Linux (L) and Xen(X)
Scalability
Conclusion

- Guest OS are modified in paravirtualization to make up for better performance
- Guest OS make Hypercalls to the hypervisor – analogous to system calls to microkernel