



Xen and the Art of Virtualization

University of Cambridge Computer Laboratory

Presenter: Jun Zhang

23/Oct/2006



Overview

- Introduction
- Xen: Approach & Overview
- Detailed Design
- Evaluation
- Related Work
- Conclusion
- Q & A



Introduction – What is Xen ?

- What is Xen ?
 - A high performance resource-managed virtual machine monitor (VMM) which enables applications such as server consolidation, co-located hosting facilities, distributed web services, secure computing platforms and application mobility.



Introduction – What is Xen ?

- What is Xen ?
 - Host commodity OSes with source code modification.
 - Enable users to dynamically instantiate an operating system to execute whatever they desire.
 - Multiplex physical resources at the granularity of an entire operating system and provide performance isolation between them.



Introduction -- Challenges

- Challenges

- Virtual machine should be isolated
- Support different OSes to accommodate the heterogeneity of popular applications
- The performance overhead of virtualization should be small



Overview

- Introduction
- Xen: Approach & Overview
- Detailed Design
- Evaluation
- Related Work
- Conclusion
- Q & A



Xen: Approach & Overview

-- Full Virtualization vs. Paravirtualization

- Full Virtualization:
 - Advantages: Do not need to modify the source code of guest OS
 - Disadvantage:
 - Drawbacks because of X86 architectural design
 - Other needs
- Paravirtualization
 - Advantages:
 - Improved performance
 - Unmodified ABI
 - Disadvantage: Need modifications to the guest OS



Xen: Approach & Overview

-- Xen Design Principles

- Support unmodified ABI.
- Supporting full multi-application OS is important.
- Paravirtualization is necessary to obtain high performance and strong resource isolation on uncooperative machine architectures.
- Even on cooperative machine architectures, completely hiding the effects of resource virtualization from guest OSes risks both correctness and performance



Xen: Approach & Overview

-- Virtual Machine Interface (Summary)

Memory Management	
Segmentation	Cannot install fully-privileged segment descriptors and cannot overlap with the top end of the linear address space.
Paging	Guest OS has direct read access to hardware page tables, but updates are batched and validated by the hypervisor. A domain may be allocated discontinuous machine pages.
CPU	
Protection	Guest OS must run at a lower privilege level than Xen.
Exceptions	Guest OS must register a descriptor table for exception handlers with Xen. Aside from page faults, the handlers remain the same.
System Calls	Guest OS may install a 'fast' handler for system calls, allowing direct calls from an application into its guest OS and avoiding indirecting through Xen on every call.
Interrupts	Hardware interrupts are replaced with a lightweight event system.
Time	Each guest OS has a timer interface and is aware of both 'real' and 'virtual' time.
Device I/O	
Network, Disk, etc.	Virtual devices are elegant and simple to access. Data is transferred using asynchronous I/O rings. An event mechanism replaces hardware interrupts for notifications.

Table 1: The paravirtualized x86 interface.



Xen: Approach & Overview

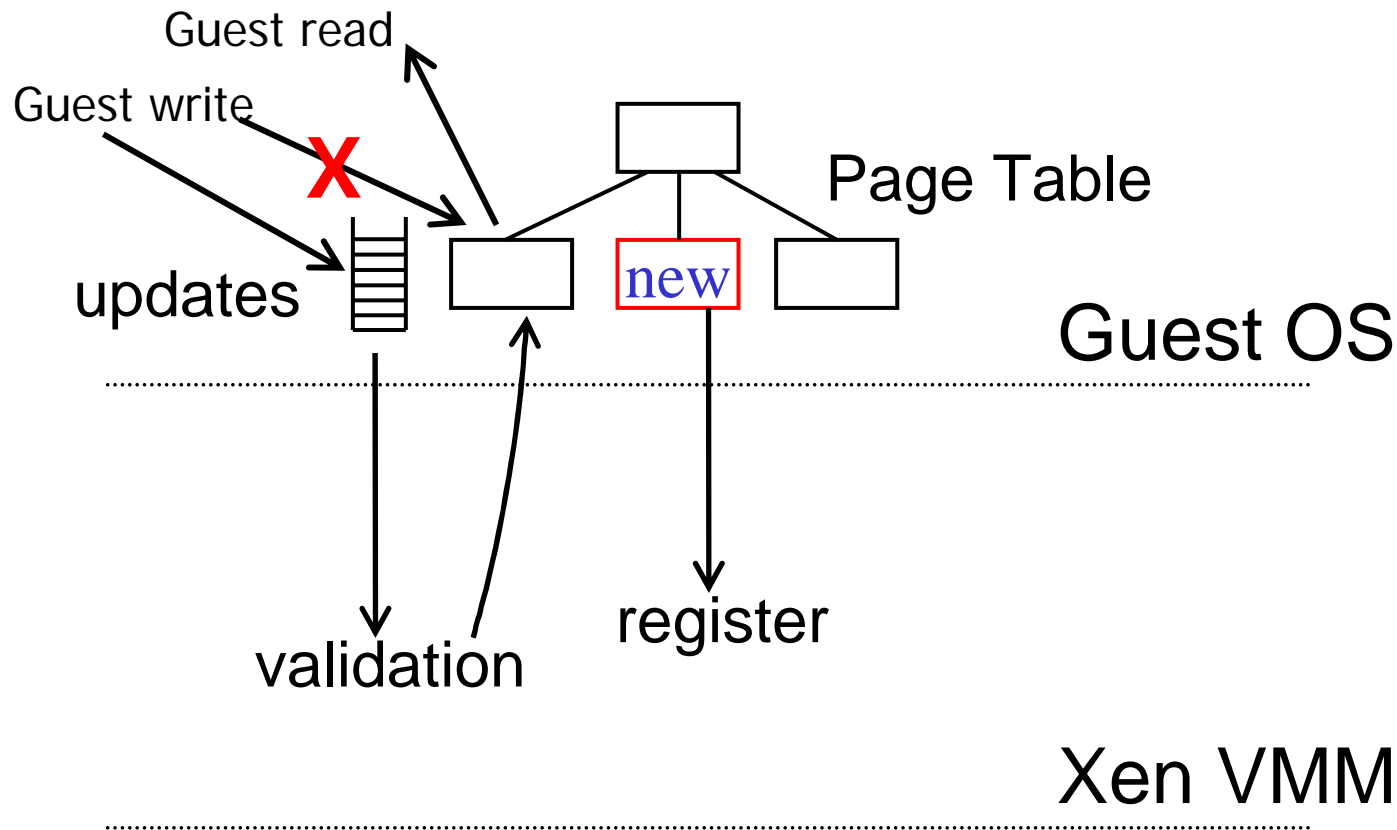
-- Virtual Machine Interface (MM)

- Hardware-managed TLB in X86 Architecture
- Memory management
 - Guest OSes allocate and manage hardware page tables
 - Xen exists in a 64MB section at the top of every address space

Xen: Approach & Overview

-- Virtual Machine Interface (MM)

■ Paging





Xen: Approach & Overview

-- Virtual Machine Interface (MM)

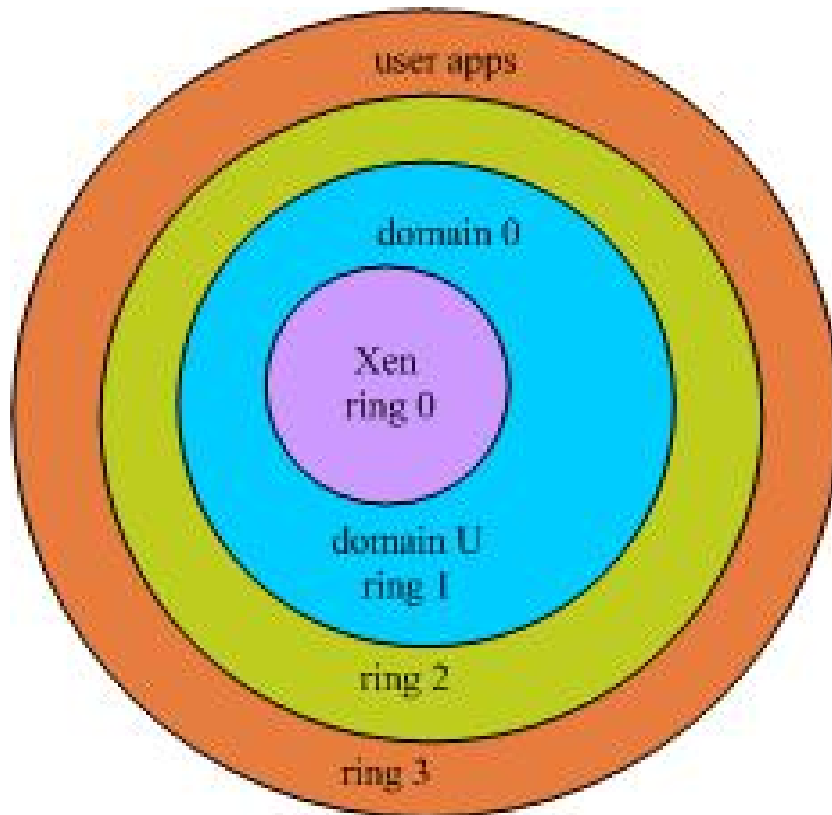
- Segmentation

- Virtualized by validating updates to hardware segment descriptor tables
- Restrictions on x86 segment descriptor tables
 - Lower privilege than Xen
 - Not allow any access to the Xen reserved portion of the address space

Xen: Approach & Overview

-- Virtual Machine Interface (CPU)

- Protection





Xen: Approach & Overview

-- Virtual Machine Interface (CPU)

- Exception
 - Register each type of exception with Xen in a table
 - The handlers specified in this table are identical to those for real X86 hardware, except for page fault handler
- System Calls
 - A 'fast' exception handler accepted by processor directly not via Xen
- Page faults
 - Must be delivered via Xen
- Handling exception
 - Validating exception handler by comparing the specified segment selector with the static values reserved for Xen
 - Any other handler problems are fixed up during exception propagation



Xen: Approach & Overview

-- Virtual Machine Interface (Device I/O)

- I/O data is transferred to and from each domain via Xen.
- Xen supports a lightweight event delivery mechanism
 - Used for sending asynchronous notification to a domain

Xen: Approach & Overview

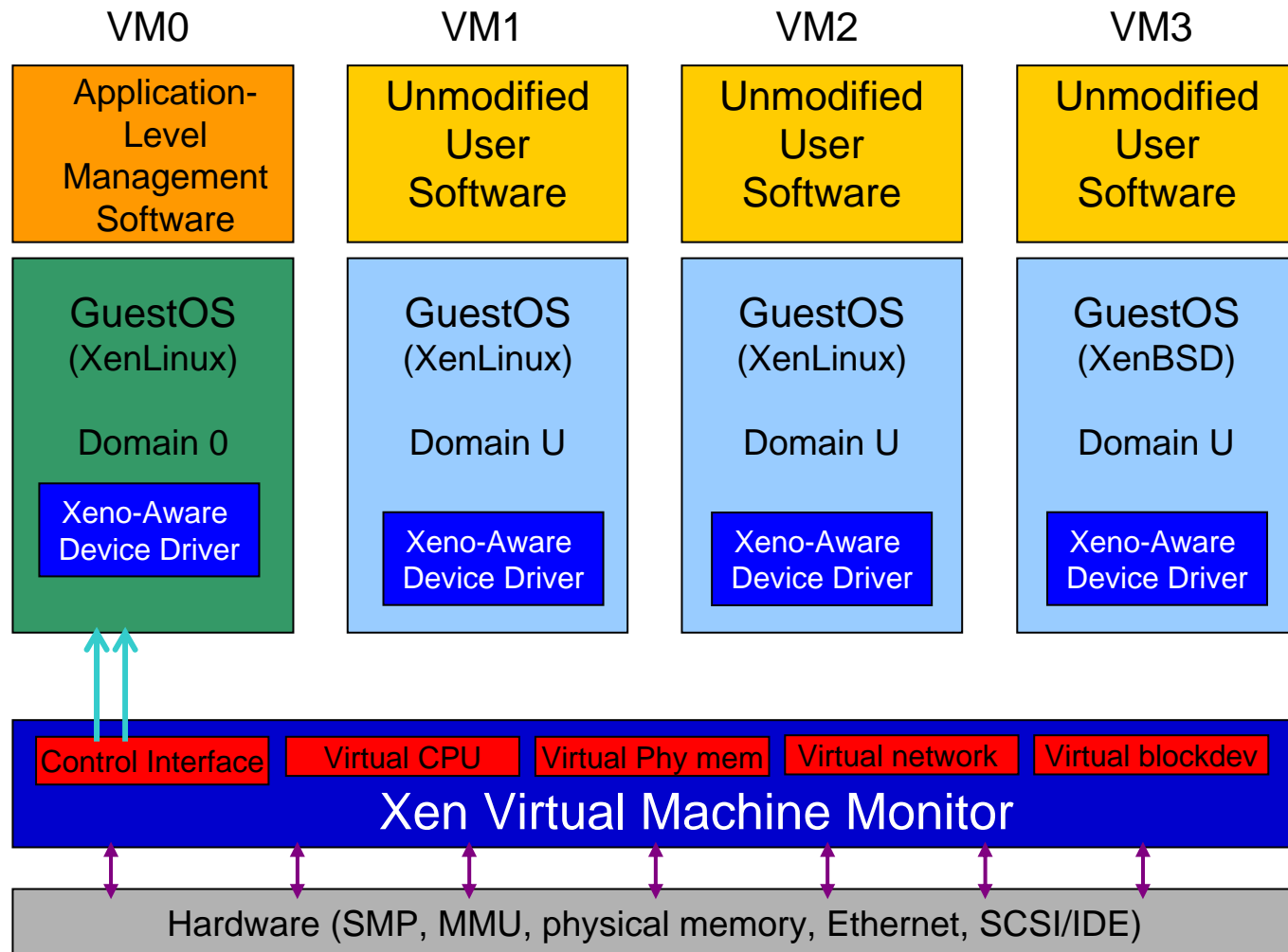
-- cost of porting an OS to Xen

OS subsection	# lines	
	Linux	XP
Architecture-independent	78	1299
Virtual network driver	484	–
Virtual block-device driver	1070	–
Xen-specific (non-driver)	1363	3321
Total	2995	4620
(Portion of total x86 code base	1.36%	0.04%)

Table 2: The simplicity of porting commodity OSes to Xen. The cost metric is the number of lines of reasonably commented and formatted code which are modified or added compared with the original x86 code base (excluding device drivers).

Xen: Approach & Overview

-- Control and Management





Overview

- Introduction
- Xen: Approach & Overview
- Detailed Design
- Evaluation
- Related Work
- Conclusion
- Q & A



Design Details

-- Control Transfer: Hypercalls and Events

- Hypercall: synchronous calls from a domain to Xen
- Asynchronous event mechanism: notifications delivered from Xen to Domain
 - Per-domain bitmask is used to store pending events

Design Details

-- Data Transfer: I/O Rings

- Main factors
 - Resource management
 - Event notification
- Structure of I/O Rings

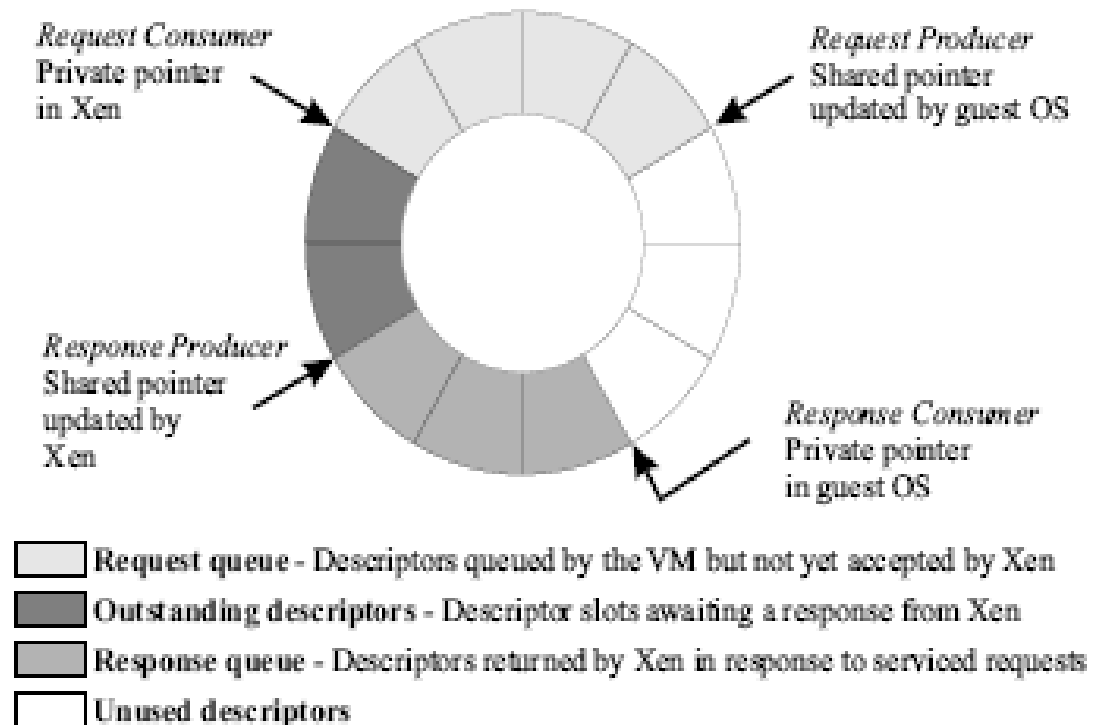


Figure 2: The structure of asynchronous I/O rings, which are used for data transfer between Xen and guest OSes.



Design Details

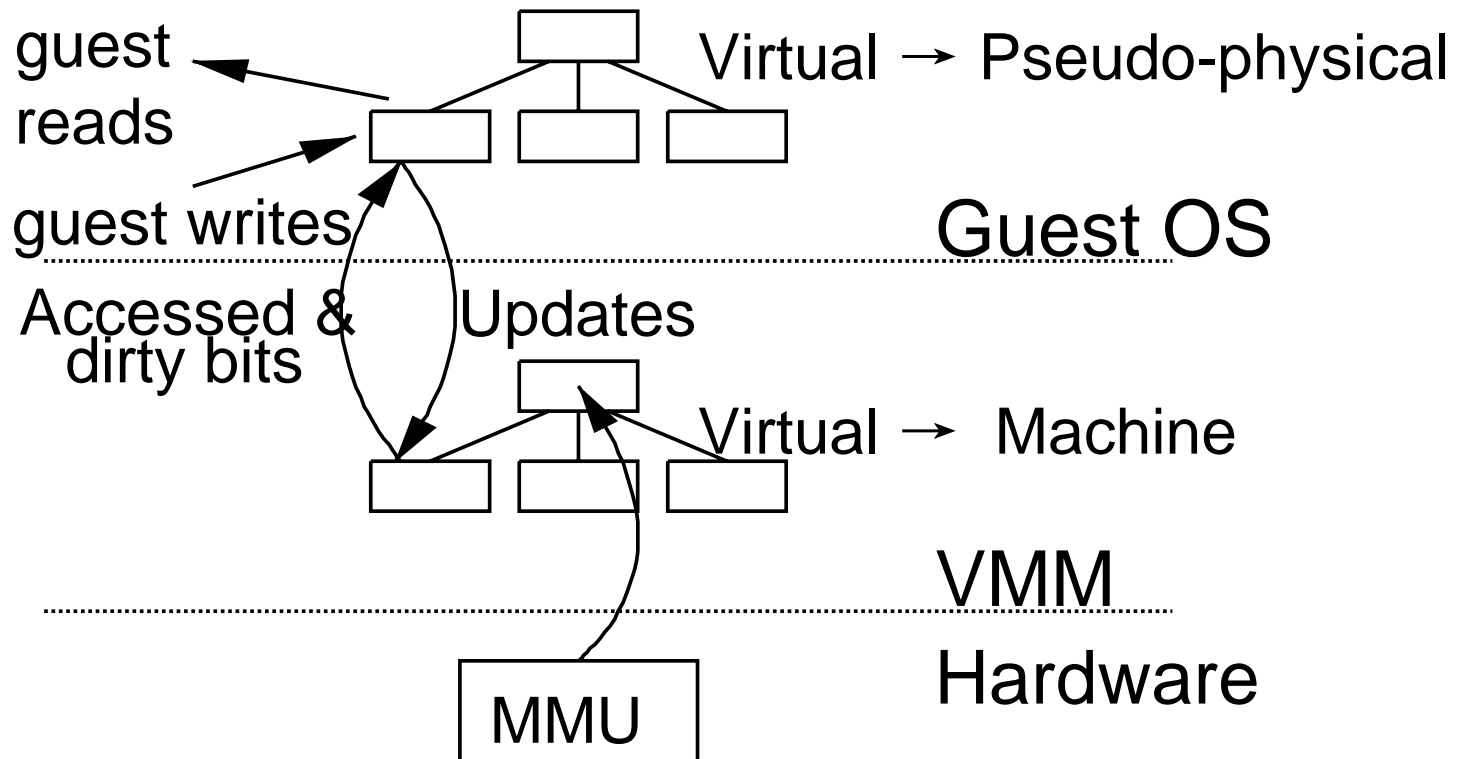
-- Subsystems Virtualization

- CPU: BTV scheduling algorithm
- real time
 - The processor's actual cycle counter
- virtual time
 - It stops when the guest OS does not occupy CPU
- wall-clock time
 - It is the real time plus an offset
- Timer
 - Alarm timers

Design Details

-- Subsystems Virtualization (Virtual Address Translation)

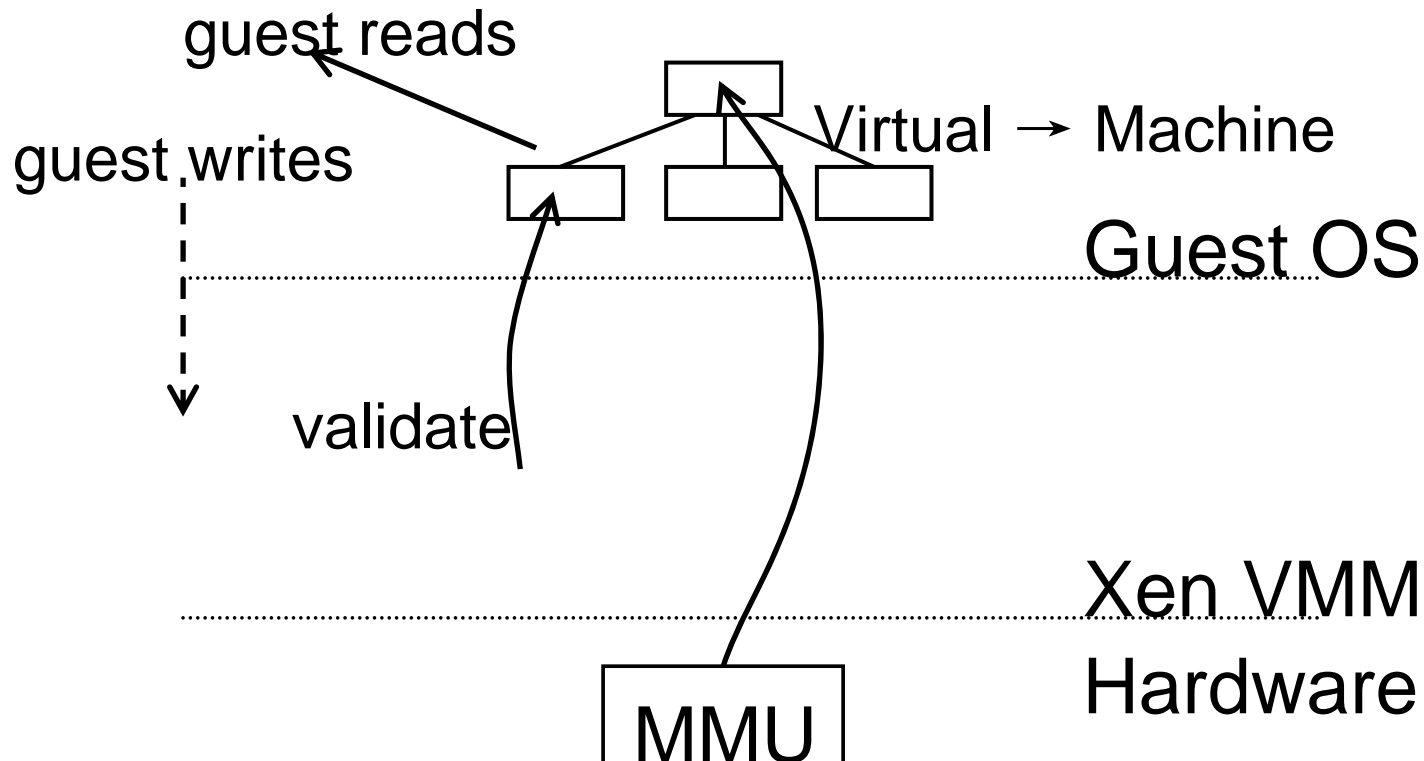
- VMware: MMU Virtualization – Shadow Mode



Design Details

-- Subsystems Virtualization (Virtual Address Translation)

- Xen: MMU Virtualization – Direct Mode



Design Details

-- Subsystems Virtualization (Virtual Address Translation)

- Page Frame:
 - Mutually-exclusively types at any point in time: PD, PT, LDT, GDT, RW
 - Maintain the invariants required for safety
 - Used to track which frames have already been validated for use in PT



Design Details

-- Subsystems Virtualization (Physical Memory)

- Memory Allocation
 - Each domain is specified when it is created
 - Maximum reservation is specified
- Balloon driver
- Illusion of contiguous physical memory
 - Not Xen but Guest OS is responsible for it



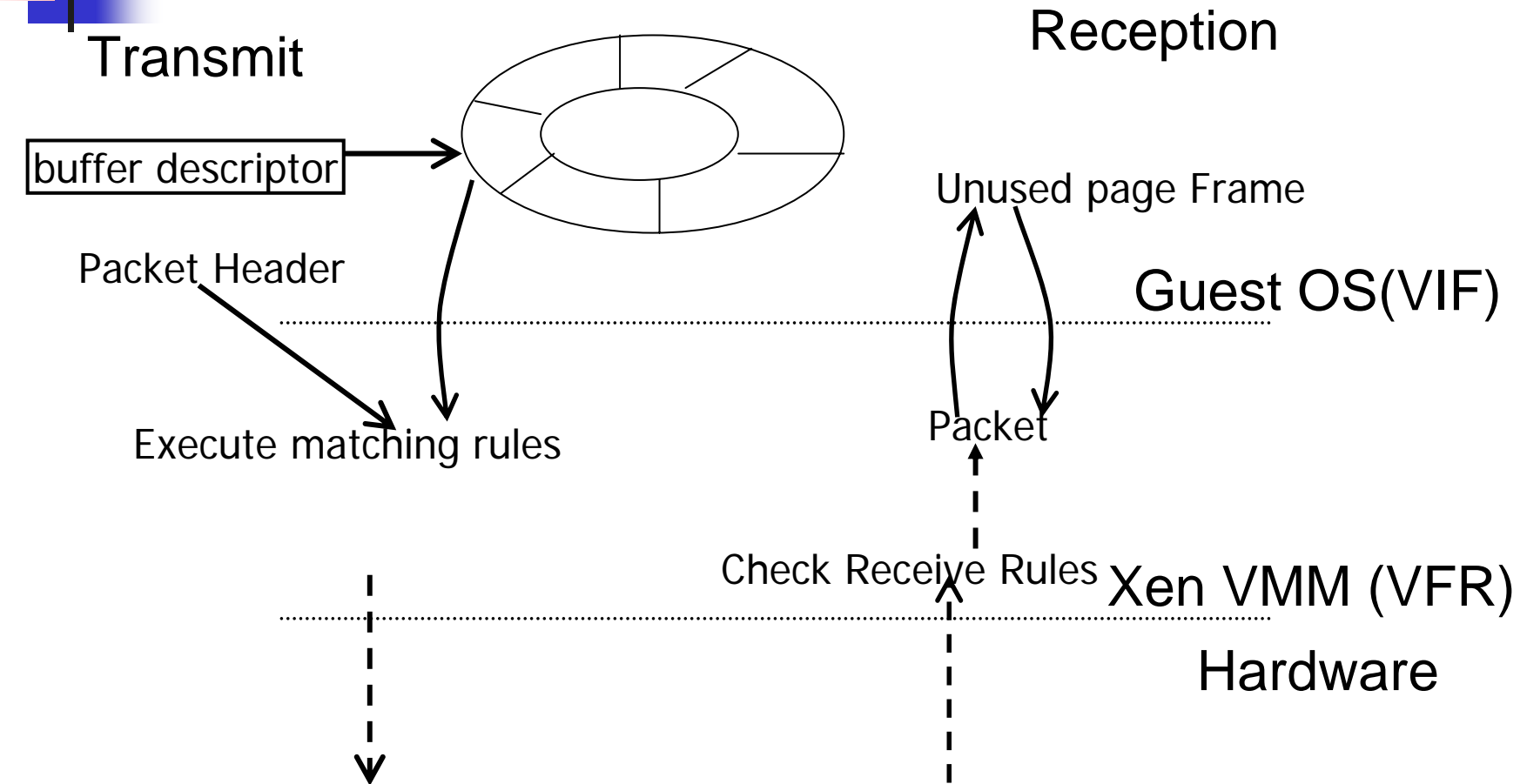
Design Details

-- Subsystems Virtualization (Network)

- VFR
- VIF
 - Like a modern network interface
 - Rules: <pattern, action>
- Packet transmission and reception

Design Details

-- Subsystems Virtualization (Network)





Design Details

-- Subsystems Virtualization (Disk)

■ VBD

- Created and configured by management software running in Domain 0
- Comprises a list of extents with associated ownership and access control information
- Accessed via I/O ring mechanism
- Guest OS and Xen scheduling algorithm
- Translation table in hypervisor, managed by Domain 0



Design Details

-- Subsystems Virtualization (build new domain)

- Domain 0 – the administration interface
 - Domain 0 is responsible for building a new domain
- Why not building a new domain entirely in Xen?
 - Reduced hypervisor complexity
 - Improved robustness
 - The ease with which the building process can be extended and specialized to cope with new guest OSes



Overview

- Introduction
- Xen: Approach & Overview
- Detailed Design
- Evaluation
- Related Work
- Conclusion
- Q & A

Evaluation

- Benchmarking of Native Linux vs. VMware Workstation 3.2 vs. Xen vs. UML

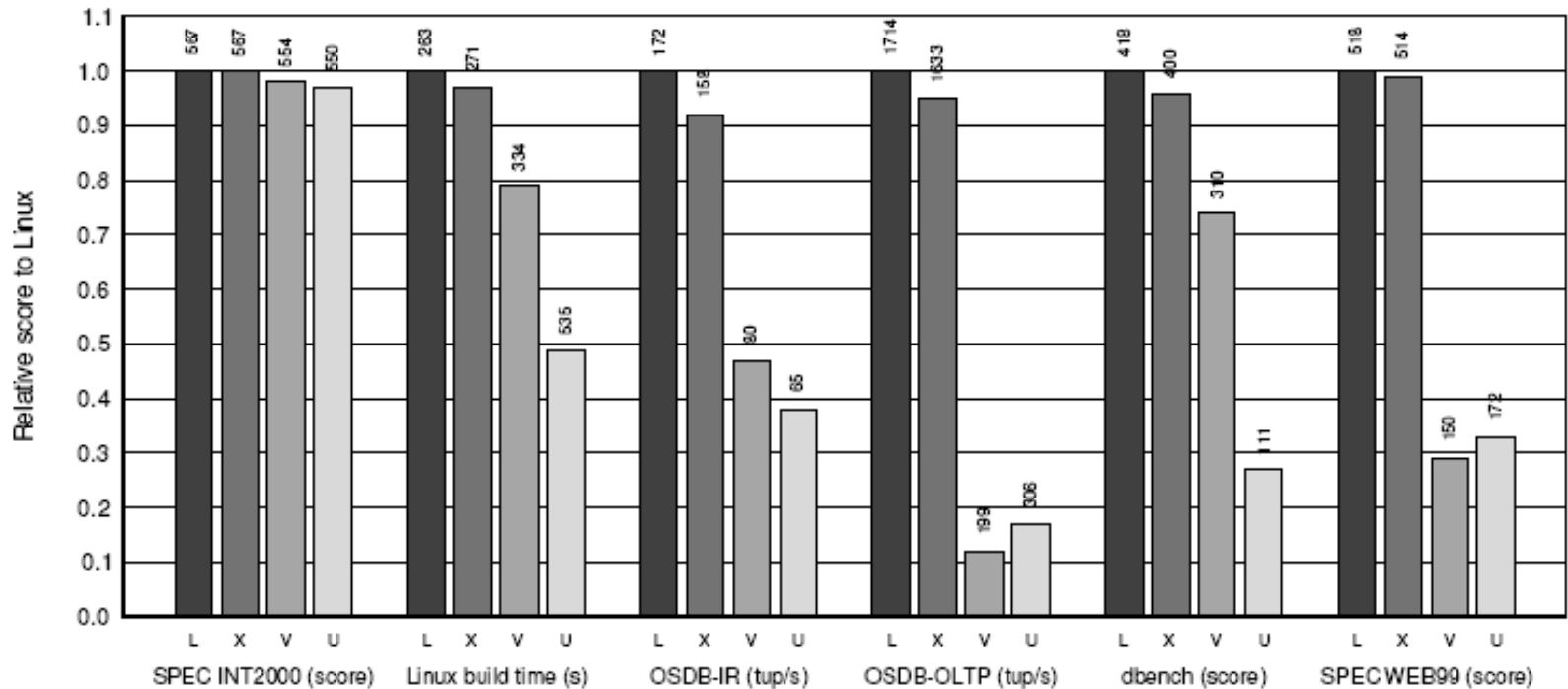


Figure 3: Relative performance of native Linux (L), XenLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

Evaluation

- Operating system benchmarks

Config	null call	null I/O	stat	opens	slct close	TCP inst	sig hndl	fork proc	exec proc	sh proc
L-SMP	0.53	0.81	2.10	3.51	23.2	0.83	2.94	143	601	4k2
L-UP	0.45	0.50	1.28	1.92	5.70	0.68	2.49	110	530	4k0
Xen	0.46	0.50	1.22	1.88	5.69	0.69	1.75	198	768	4k8
VMW	0.73	0.83	1.88	2.99	11.1	1.02	4.63	874	2k3	10k
UML	24.7	25.1	36.1	62.8	39.9	26.0	46.0	21k	33k	58k

Table 3: `lmbench`: Processes - times in μs

Config	0K File		10K File		Mmap lat	Prot fault	Page fault
	create	delete	create	delete			
L-SMP	44.9	24.2	123	45.2	99.0	1.33	1.88
L-UP	32.1	6.08	66.0	12.5	68.0	1.06	1.42
Xen	32.5	5.86	68.2	13.6	139	1.40	2.73
VMW	35.3	9.3	85.6	21.4	620	7.53	12.4
UML	130	65.7	250	113	1k4	21.8	26.3

Table 5: `lmbench`: File & VM system latencies in μs

Config	2p	2p	2p	8p	8p	16p	16p
	0K	16K	64K	16K	64K	16K	64K
L-SMP	1.69	1.88	2.03	2.36	26.8	4.79	38.4
L-UP	0.77	0.91	1.06	1.03	24.3	3.61	37.6
Xen	1.97	2.22	2.67	3.07	28.7	7.08	39.4
VMW	18.1	17.6	21.3	22.4	51.6	41.7	72.2
UML	15.5	14.6	14.4	16.3	36.8	23.6	52.0

Table 4: `lmbench`: Context switching times in μs



Evaluation

- Network Performance

	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

Evaluation

- Running multiple applications in own guest OS vs. Running those applications on native OS.

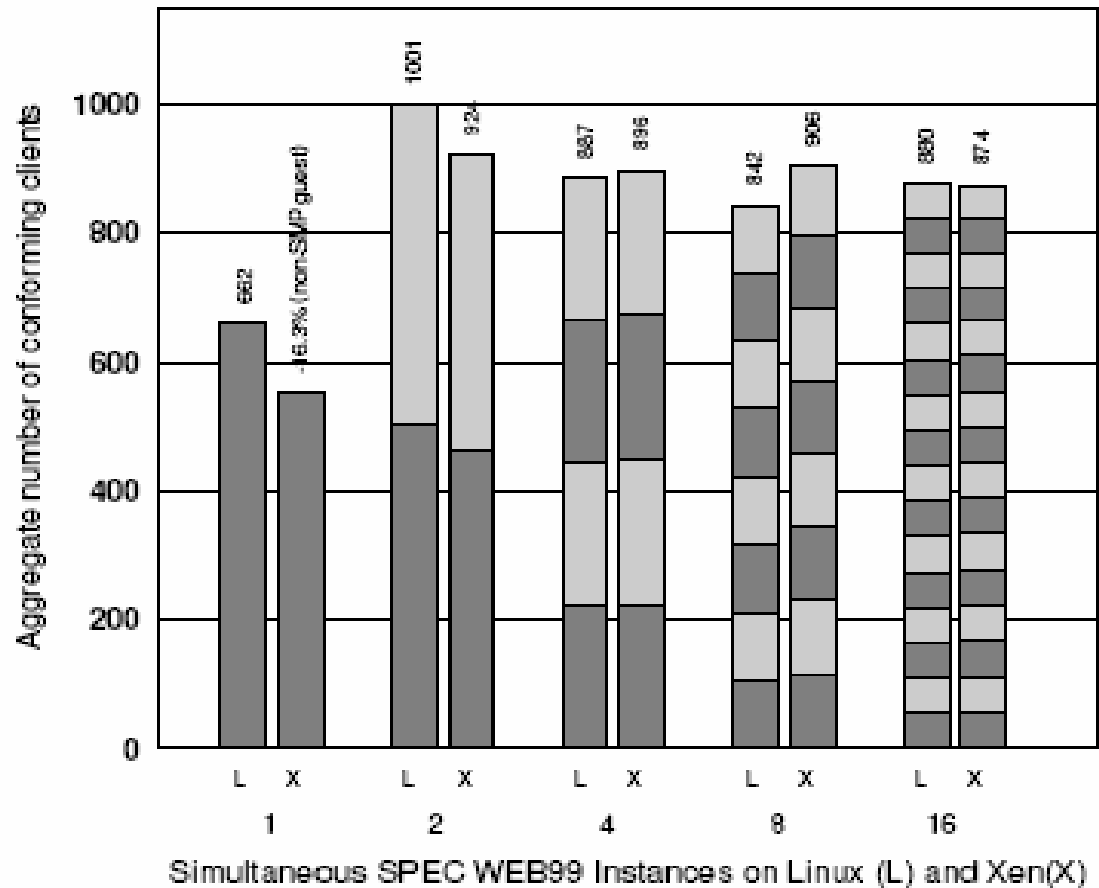


Figure 4: SPEC WEB99 for 1, 2, 4, 8 and 16 concurrent Apache servers: higher values are better.



Evaluation

- Performance isolation
 - No comparative evaluation here
 - Testing:
 - Two domains run PostgreSQL/OSDB-IR and SPECWEB99
 - One domain runs dd
 - One domain runs fork-bomb

Evaluation

- Scalability: examine Xen's ability to scale to its target of 100 domains

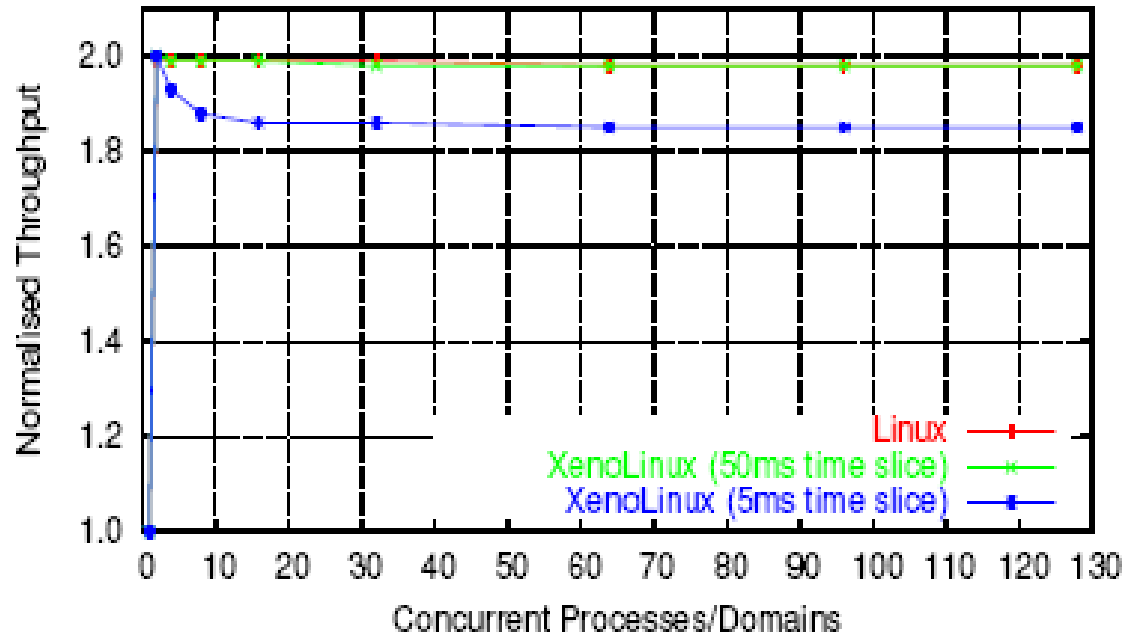


Figure 6: Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains



Overview

- Introduction
- Xen: Approach & Overview
- Detailed Design
- Evaluation
- Related Work
- Conclusion
- Q & A



Related Work

- Resource Containers
 - E.g. Solaris Zones
- Full Virtualization:
 - IBM VM/370; VMware; Connectix; Disco
- Paravirtualization:
 - A paravirtualization Linux for IBM zSeries Mainframe; Denali
- Low-level virtualization application
 - vMatrix; PlanetLab



Related Work – Full Virtualization

- VMWare Solutions
 - VMWare Workstation (Hosted Architecture)
 - The VM and the guest OS are installed and run on the top of a standard commodity OS
 - VMWare ESX Server (Hypervisor Architecture)
 - Installs a layer of software, hypervisor, directly on top of the bare hardware and guest OSes runs on top of the hypervisor



Related Work – VMWare vs. Xen

- VMWare

- While developing some features relative to kernel, we had better choose VMWare
- More mature
- Cost a lot of money (VMWare ESX Server)

- Xen

- When performance is the most important factor, we had better use Xen.
- Open Source
- More and more commodity OSes support it



Related Work

-- ParaVirtualization

- Denali

- Designed for small-scale and unpopular network services

- Denali vs. Xen

- Not target existing ABIs
- Not address the problem of supporting application multiplexing, nor multiple address spaces within a single guest OS
- No performance isolation
- Virtualizes the 'namespaces' of all machine resources



Overview

- Introduction
- Xen: Approach & Overview
- Detailed Design
- Evaluation
- Related Work
- Conclusion
- Q & A



Conclusions

- Xen do have excellent performance
- Xen is open source which cost little money
- Now more and more commodity OSes begin to support it.

It has great potential in future!



Q&A

- It seems like both Intel(Vanderpool) and AMD have recently launched their x86 products with virtualization support. Even though the virtualization idea was around for quite some time is it actually this Xen architecture which persuaded the vendors?
- Recently Intel and AMD included new hardware supported virtualization in their latest product, which allows current OSes to be virtualized without any modification. Does this render paravirtualization meaningless? (I do see Xen switching to hardware supported virtualization in their 3.0 release.)



Q&A

- What are the advantages and innovative features of this approach compared to the approach of the emulation of VMware?
- Is the reservation of the top 64 MB of each address space really as non-problematic as the paper makes it out to be? They mention the 'common ABIs' not requiring them, but Murphy's law dictates that the systems that Xen will be running under will be running apps that uses this top addy space almost continuously.



Q&A

- What is the role of Domain0 in Xen? Why is not implemented as part of Xen? Does this introduce a security issue?
- I was somewhat disappointed that the authors did not discuss how scheduling algorithms for guest OSes differs from scheduling ordinary processes. Why do you think the authors avoided this question? Do we expect that there would be unique challenges in using the entire running OS as a unit of scheduling?



Q&A

- Is Xen in its current form scalable to the cluster level? I hope it is! If so, how does Xen interact with its mirrors on other machines in a cluster environment? Is domain 0/hypervisor scalable? Is there a possibility of large scale full/para virtualization on the cluster level when implementing Xen or does it run on individual machines alone and supports interaction between different virtual machine instances? What will be the sub-instances in that case? How can large applications like Brain Modeling, N-Body be mapped simultaneously along with minor applications on a Xen Cluster environment efficiently?