VMMS: DISCO AND XEN

CS6410

Ken Birman

Disco (First version of VMWare)

Edouard Bugnion, Scott Devine, and Mendel Rosenblum

Virtualization

- *"a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications, or end users interact with those resources. This includes making a single physical resource appear to function as multiple logical resources; or it can include making multiple physical resources appear as a single logical resource"*

Old idea from the 1960s

- IBM VM/370 – A VMM for IBM mainframe
  - Multiple OS environments on expensive hardware
  - Desirable when few machine around
- Popular research idea in 1960s and 1970s
  - Entire conferences on virtual machine monitors
  - Hardware/VMM/OS designed together
- Interest died out in the 1980s and 1990s
  - Hardware got more cheaper
  - Operating systems got more powerful (e.g. multi-user)

A Return to Virtual Machines

- Disco: Stanford research project (SOSP ’97)
  - Run commodity OSes on scalable multiprocessors
  - Focus on high-end: NUMA, MIPS, IRIX
- Commercial virtual machines for x86 architecture
  - VMware Workstation (now EMC) (1999-)
  - Connectix VirtualPC (now Microsoft)
- Research virtual machines for x86 architecture
  - Xen (SOSP ’03)
  - plex86
- OS-level virtualization
  - FreeBSD Jails, User-mode-linux, UMLinux

Overview

- Virtual Machine
  - *A fully protected and isolated copy of the underlying physical machine's hardware. (definition by IBM)"*
- Virtual Machine Monitor
  - *A thin layer of software that's between the hardware and the Operating system, virtualizing and managing all hardware resources.*
  - Also known as "Hypervisor"

Classification of Virtual Machines

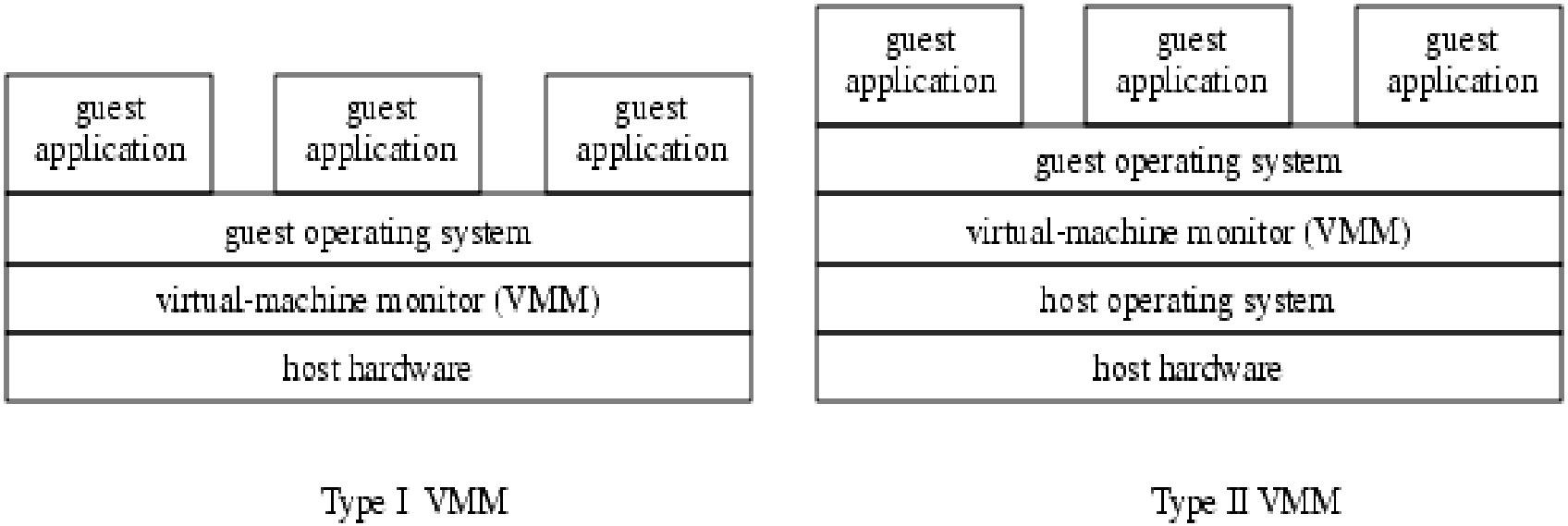


Figure 1: Virtual-machine structures. A virtual-machine monitor is a software layer that runs on a host platform and provides an abstraction of a complete computer system to higher-level software. The host platform may be the bare hardware (Type I VMM) or a host operating system (Type II VMM). The software running above the virtual-machine abstraction is called guest software (operating system and applications).

Classification of Virtual Machines

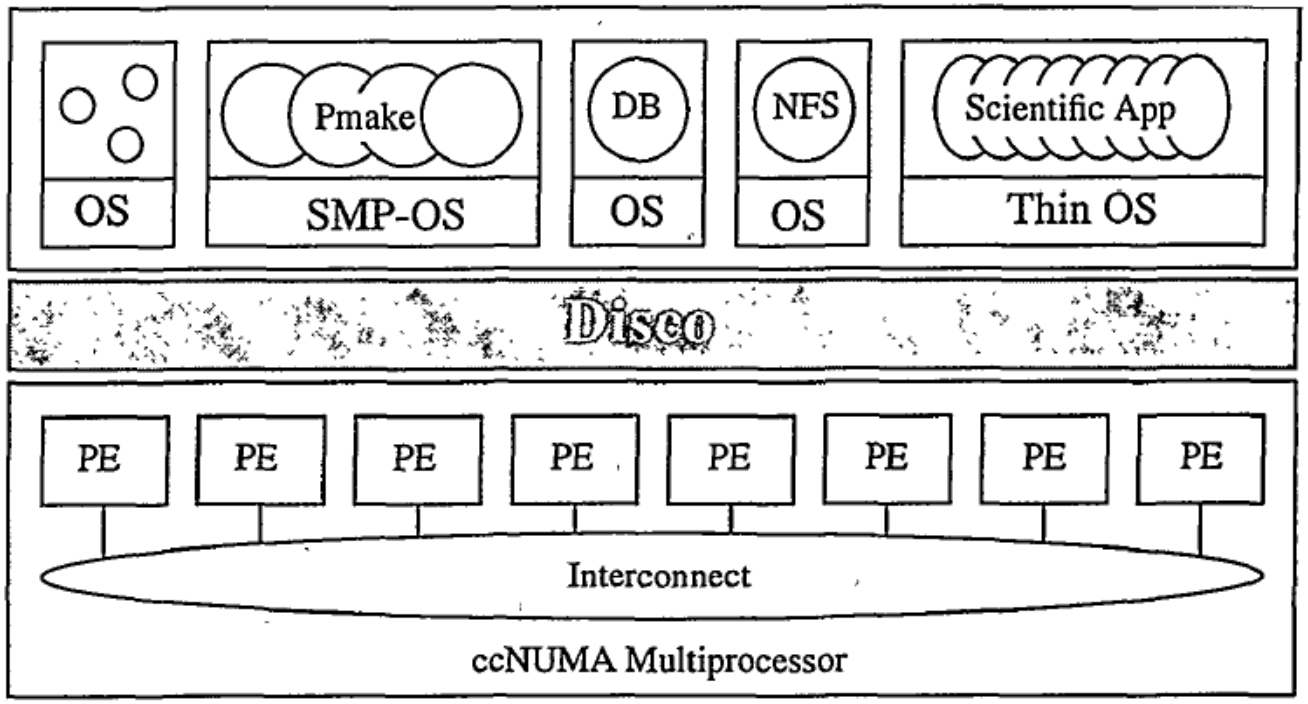
- Type I
  - VMM is implemented directly on the physical hardware.
  - VMM performs the scheduling and allocation of the system's resources.
  - IBM VM/370, Disco, VMware's ESX Server, Xen
- Type II
  - VMMs are built completely on top of a host OS.
  - The host OS provides resource allocation and standard execution environment to each "guest OS."
  - User-mode Linux (UML), UMLinux

Disco: Challenges

- Overheads
  - Additional Execution
  - Virtualization I/O
  - Memory management for multiple VMs
- Resource Management
  - Lack of information to make good policy decisions
- Communication & Sharing
  - Interface to share memory between multiple VMs

Baseed on slides from 2011fa: Ashik R

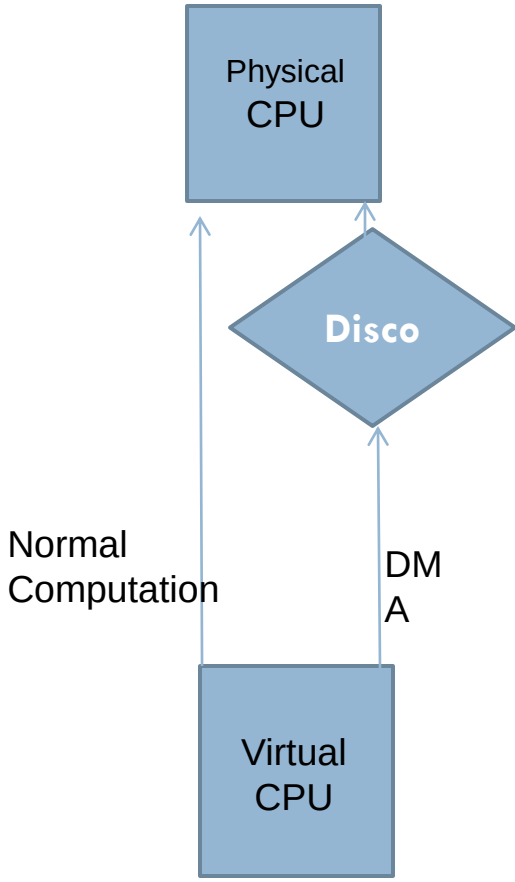
Disco: Interface



- Processors – Virtual CPU
- Memory
- I/O Devices

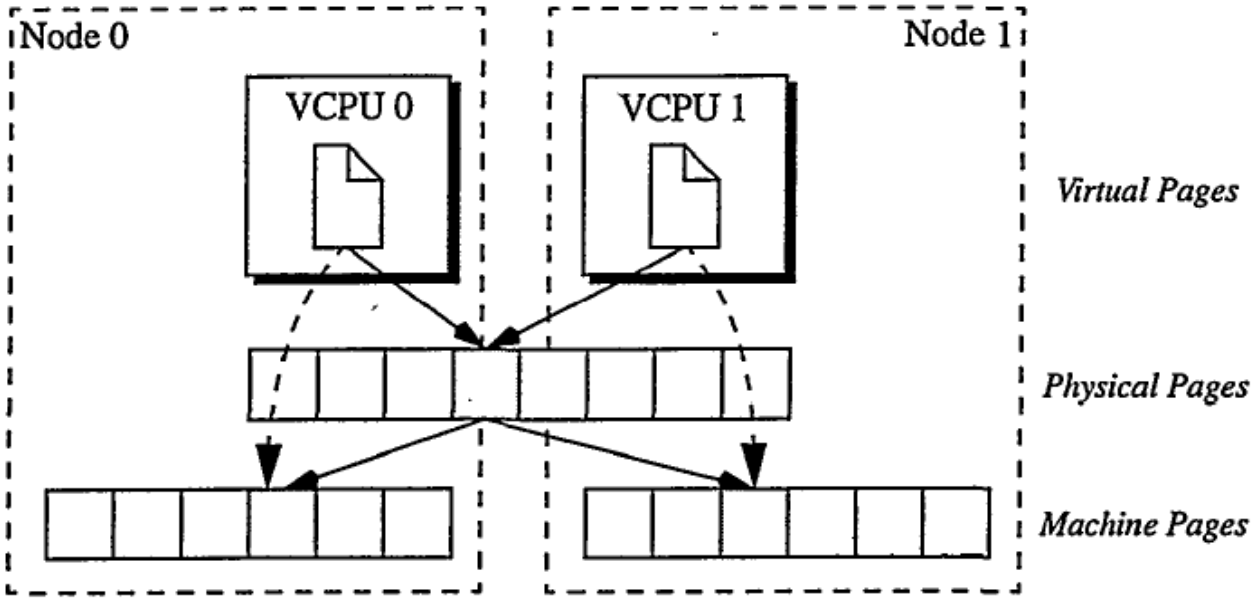
Disco: Virtual CPUs

- Direct Execution on the real CPU
- Intercept Privileged Instructions
- Different Modes:
  - Kernel Mode: Disco
  - Supervisor Mode: Virtual Machines
  - User Mode: Applications



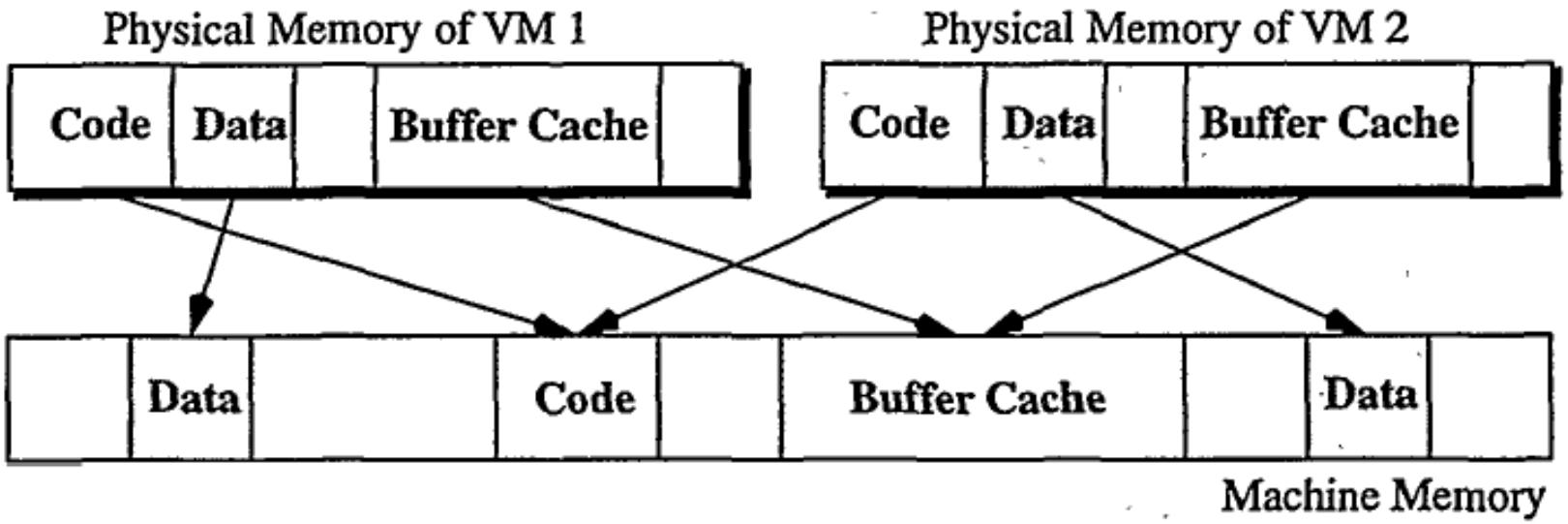
Disco: Memory Virtualization

- Adds a level of address translation
- Uses Software reloaded TLB and pmap
- Flushes TLB on VCPU Switch
- Uses second level Software TLB



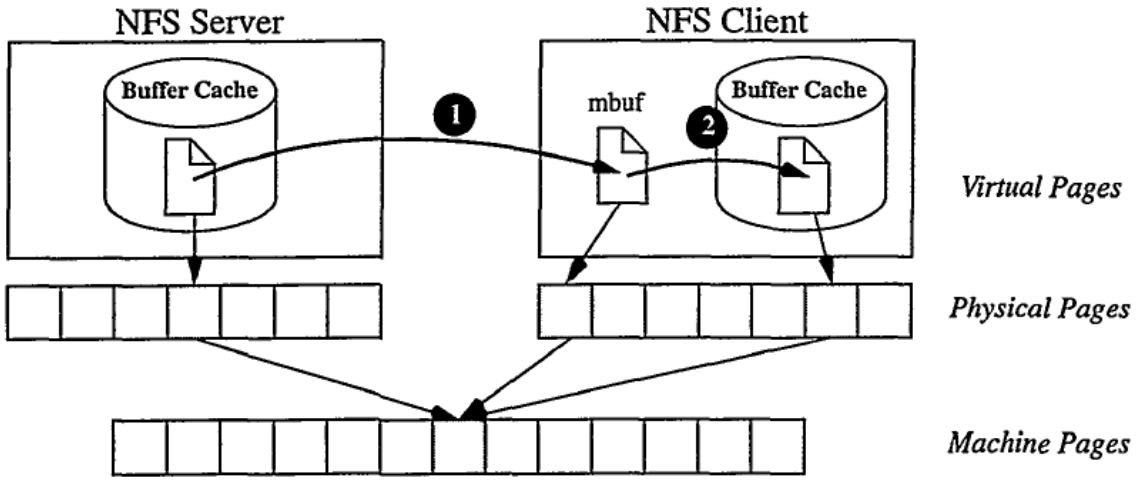
Disco: Memory Management

- Affinity Scheduling
- Page Migration
- Page Replication
- memmap



Disco: I/O Virtualization

- Virtualizes access to I/O devices and intercepts all device access
- Adds device drivers in to OS
- Special support for Disk and Network access
  - Copy-on-write
  - Virtual Subnet
- Allows memory sharing between VMs agnostic of each other



Running Commodity OSes

- Changes for MIPS Architecture
  - Required to relocate the unmapped segment
- Device Drivers
  - Added device drivers for I/O devices.
- Changes to the HAL
  - Inserted some monitor calls in the OS

Experimental Results

- Uses Sim OS Simulator for Evaluations

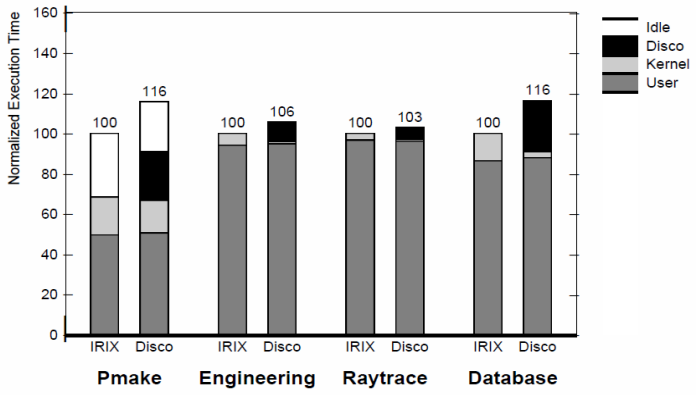


Fig. 6. Overhead of virtualization

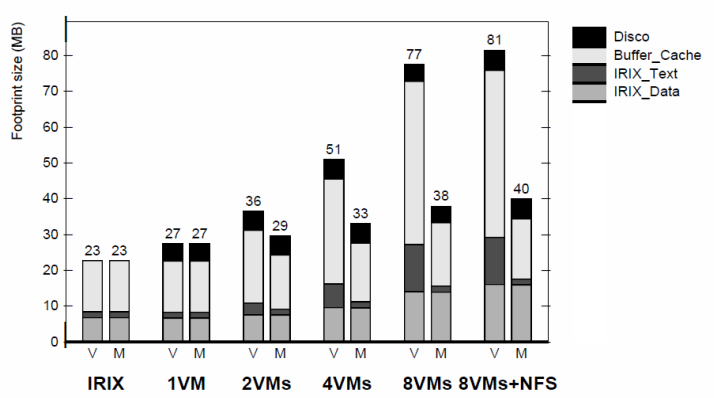


Fig. 7. Data sharing in Disco between virtual machines

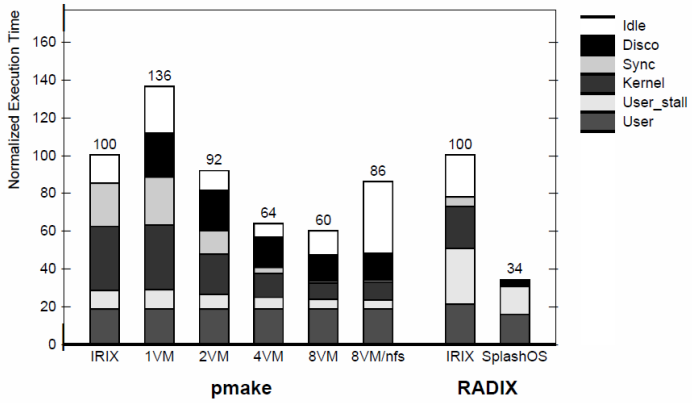


Fig. 8. Workload scalability under Disco

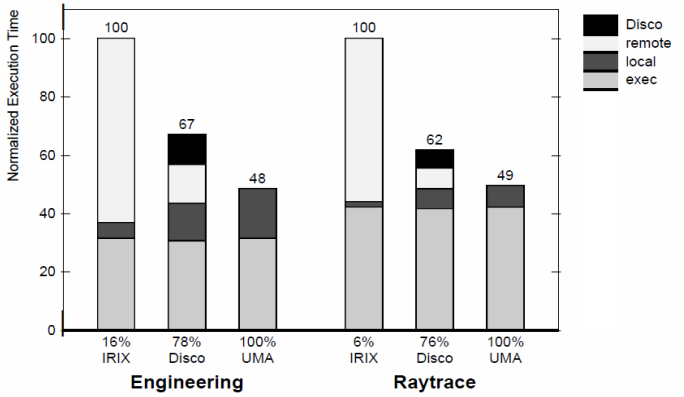


Fig. 9. Performance benefits of page migration and replication

Disco: Takeaways

- Develop system s/w with less effort
- Low/Modest overhead
- Simple solution for Scalable Hardware

- Subsequent history
  - Rewritten into VMWare, became a major product
  - Performance hit a subject of much debate but successful even so, and of course evolved greatly
  - Today a huge player in cloud market

XEN AND THE ART OF VIRTUALIZATION

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris,

Alex Ho, Rolf Neugebauery, Ian Pratt, Andrew Wareld

Xen's Virtualization Goals

- Isolation
- Support different Operating Systems
- Performance overhead should be small

Reasons to Virtualize

- Systems hosting multiple applications on a shared machine
- undergo the following problems:
- Do not support adequate isolation
- Affect of Memory Demand, Network Traffic, Scheduling Priority and Disk Access on process's performance
- System Administration becomes Difficult

XEN : Introduction

- A Para-Virtualized Interface
- Can host Multiple and different Operating Systems
- Supports Isolation
- Performance Overhead is minimum
- Can Host up to 100 Virtual Machines

XEN : Approach

- Drawbacks of Full Virtualization with respect to x86 architecture
  - Support for virtualization not inherent in x86 architecture
  - Certain privileged instructions did not trap to the VMM
  - Virtualizing the MMU efficiently was difficult
  - Other than x86 architecture deficiencies, it is sometimes required to view the real and virtual resources from the guest OS point of view

- Xen's Answer to the Full Virtualization problem:
  - It presents a virtual machine abstraction that is similar but not identical to the underlying hardware -para-virtualization
  - Requires Modifications to the Guest Operating System
  - No changes are required to the Application Binary Interface (ABI)

Terminology Used

- Guest Operating System (OS) – refers to one of the operating systems that can be hosted by XEN.
- Domain – refers to a virtual machine within which a Guest OS runs and also an application or applications.
- Hypervisor – XEN (VMM) itself.

XEN's Virtual Machine Interface

- The virtual machine interface can be broadly
- classified into 3 parts. They are:
- Memory Management
- CPU
- Device I/O

XEN's VMI : Memory Management

- Problems
  - x86 architecture uses a hardware managed TLB
  - Segmentation
- Solutions
  - One way would be to have a tagged TLB, which is currently supported by some RISC architectures
  - Guest OS are held responsible for allocating and managing the hardware page tables but under the control of Hypervisor
  - XEN should exist (64 MB) on top of every address space
- Benefits
  - Safety and Isolation
  - Performance Overhead is minimized

XEN's VMI : CPU

- Problems
  - Inserting the Hypervisor below the Guest OS means that the Hypervisor will be the most privileged entity in the whole setup
  - If the Hypervisor is the most privileged entity then the Guest OS has to be modified to execute in a lower privilege level
  - Exceptions

- Solutions
  - x86 supports 4 distinct privilege levels – rings
  - Ring 0 is the most and Ring 3 is the least
  - Allowing the guest OS to execute in ring 1- provides a way to catch the
  - privileged instructions of the guest OS at the Hypervisor
  - Exceptions such as memory faults and software traps are solved by registering the handlers with the Hypervisor
  - Guest OS must register a fast handler for system calls with the Hypervisor
  - Each guest OS will have their own timer interface

XEN's VMI: Device I/O

- Existing hardware Devices are not emulated
- A simple set of device abstractions are used – to ensure protection and isolation
- Data is transferred to and fro using shared memory, asynchronous buffer descriptor rings – performance is better
- Hardware interrupts are notified via a event delivery mechanism to the respective domains

XEN : Cost of Porting Guest OS

- Linux is completely portable on the Hypervisor - the OS is called XenoLinux
- Windows XP is in the Process
- Lot of modifications are required to the XP's architecture Independent code – lots of structures and unions are used for PTE's
- Lot of modifications to the architecture specific code was done in both the OSes
- In comparing both OSes – Larger Porting effort for XP

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 : Control and Management

- Xen exercises just basic control operations such as access control, CPU scheduling between domains etc.
- All the policy and control decisions with respect to Xen are undertaken by management software running on one of the domains – domain0
- The software supports creation and deletion of VBD, VIF, domains, routing rules etc.

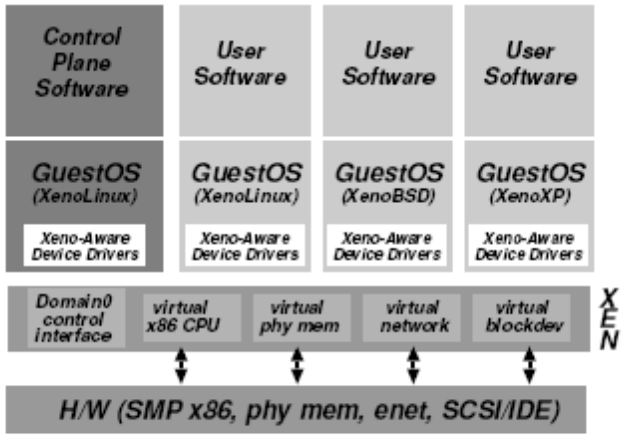


Figure 1: The structure of a machine running the Xen hypervisor, hosting a number of different guest operating systems, including *Domain0* running control software in a XenoLinux environment.

XEN : Detailed Design

- Control Transfer
  - Hypercalls – Synchronous calls made from domain to XEN
  - Events – Events are used by Xen to notify the domain in an asynchronous manner
- Data Transfer
  - Transfer is done using I/O rings
  - Memory for device I/O is provided by the respective domain
  - Minimize the amount of work to demultiplex data to a specific domain

XEN : Data Transfer in Detail

- I/O Ring Structure

- I/O Ring is a circular queue of descriptors
- Descriptors do not contain I/O data but indirectly reference a data buffer as allocated by the guest OS.
- Access to each ring is based on a set of pointers namely producer and consumer pointers
- Guest OS associates a unique identifier with each request, which is replicated by the response to address the possible problem of ordering between requests

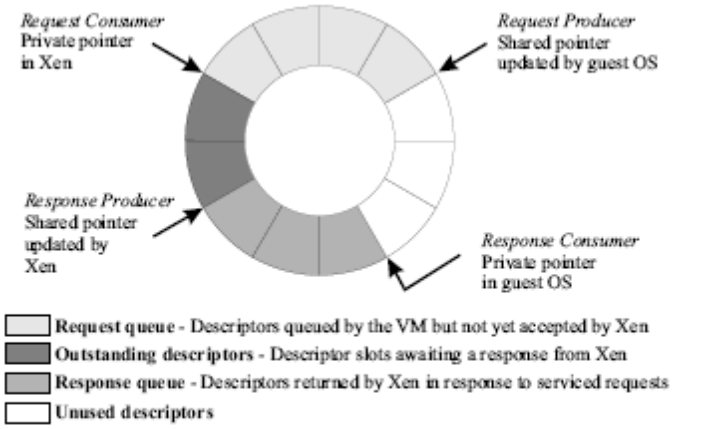


Figure 2: The structure of asynchronous I/O rings, which are used for data transfer between Xen and guest OSes.

XEN : Sub System Virtualization

- The various Sub Systems are :
- CPU Scheduling
- Time and Timers
- Virtual Address Translation
- Physical Memory
- Network Management
- Disk Management

XEN : CPU Scheduling

- Xen uses Borrowed Virtual Time scheduling
- algorithm for scheduling the domains
- Per domain scheduling parameters can be adjusted using domain0

- Advantages

- Work – Conserving
- Low – Latency Dispatch by using virtual time warping

XEN : Time and Timers

- Guest OSes are provided information about real time, virtual time and wall clock time
- Real Time – Time since machine boot and is accurately maintained with respect to the processor's cycle counter and is expressed in nanoseconds
- Virtual Time – This time is increased only when the domain is executing – to ensure correct time slicing between application processes on its domain
- Wall clock Time – an offset that can be added to the current real time.

XEN : Virtual Address Translation

- Register guest OSes page tables directly with the MMU
- Restrict Guest OSes to Read only access
- Page table Updates should be validated through the hypervisor to ensure safety
- Each page frame has two properties associated with it namely type and reference count
- Each page frame at any point in time will have just one of the 5 mutually exclusive types:
  - Page directory (PD), page table (PT), local descriptor table (LDT), global descriptor table (GDT), or writable (RW).
- A page frame is allocated to page table use after validation and it is pinned to PD or PT type.
- A frame can't be re-tasked until reference=0 and it is unpinned.
- To minimize overhead of the above operations in a batch process.
- The OS fault handler takes care of frequently checking for updates to the shadow page table to ensure correctness.

XEN : Physical Memory

- Physical Memory Reservations or allocations are made at the time of creation which are statically partitioned, to provide strong isolation.
- A domain can claim additional pages from the hypervisor but the amount is limited to a reservation limit.
- Xen does not guarantee to allocate contiguous regions of memory, guest OSes will create the illusion of contiguous physical memory.
- Xen supports efficient hardware to physical address mapping through a shared translation array, readable by all domains – updates to this are validated by Xen.

XEN : Network Management

- Xen provides the abstraction of a virtual firewall router (VFR), where each domain has one or more Virtual network interface (VIF) logically attached to this VFR.
- The VIF contains two I/O rings of buffer descriptors, one for transmitting and the other for receiving
- Each direction has a list of associated rules of the form (<pattern>,<action>) – if the pattern matches then the associated action applied.
- Domain0 is responsible for implementing the rules over the different domains.
- To ensure fairness in transmitting packet they implement round-robin packet scheduler.

XEN : Disk Management

- Only Domain0 has direct unchecked access to the physical disks.
- Other Domains access the physical disks through virtual block devices (VBDs) which is maintained by domain0.
  - VBS comprises a list of associated ownership and access control information, and is accessed via I/O ring.
- A translation table is maintained for each VBD by the hypervisor, the entries in the VBD's are controlled by domain0.
- Xen services batches of requests from competing domains in a simple round-robin fashion.

XEN : Building a New Domain

- Building initial guest OS structures for new domains is done by domain0.
- Advantages are reduced hypervisor complexity and improved robustness.
- The building process can be extended and specialized to cope with new guest OSes.

XEN : EVALUATION

- Different types of evaluations:
  - Relative Performance.
  - Operating system benchmarks.
  - Concurrent Virtual Machines.
  - Performance Isolation.
  - Scalability

XEN : Experimental Setup

- Dell 2650 dual processor 2.4GHz Xeon server with 2GB RAM
- A Broadcom Tigon 3 Gigabit Ethernet NIC.
- A single Hitachi DK32EJ 146GB 10k RPM SCSI disk.
- Linux Version 2.4.21 was used throughout, compiled for architecture for native and VMware guest OS experiments– i686
- Xeno-i686 architecture for Xen.
- Architecture um for UML (user mode Linux)
- The products to be compared are native Linux (L), XenoLinux (X), VMware Workstation 3.2 (V) and User Mode Linux (U)

XEN : Relative Performance

- Complex application-level benchmarks that exercise the whole system have been employed to characterize performance.
- First suite contains a series of long-running computationally-intensive applications to measure the performance of system's processor, memory system and compiler quality.
  - Almost all execution are all in user-space, all VMMs exhibit low overhead.
- Second, the total elapsed time taken to build a default configuration of the Linux 2.4.21 kernel on a local ext3 file system with gcc 2.96
  - Xen – 3% overhead, others more significant slowdown.
- Third and fourth, experiments performed using PostgreSQL 7.1.3 database, exercised by the Open Source Database Benchmark Suite (OSDB) for multi-user Information Retrieval (IR) and On-Line Transaction Processing (OLTP) workloads
  - PostgreSQL places considerable load on the operating system which leads to substantial virtualization overheads on VMware and UML.

XEN : Relative Performance

- Fifth, dbench program is a file system benchmark derived from 'NetBench'
  - Throughput experienced by a single client performing around 90,000 file system operations.
- Sixth, a complex application-level benchmark for evaluating web servers and the file systems
  - 30% are dynamic content generation, 16% are HTTP POST operations and 0.5% execute a CGI script. There is up to 180Mb/s of TCP traffic and disk activity on 2GB dataset.
  - XEN fares well with 1% performance of native Linux, VMware and UML less than a third of the number of clients of the native Linux system.

XEN : Relative Performance

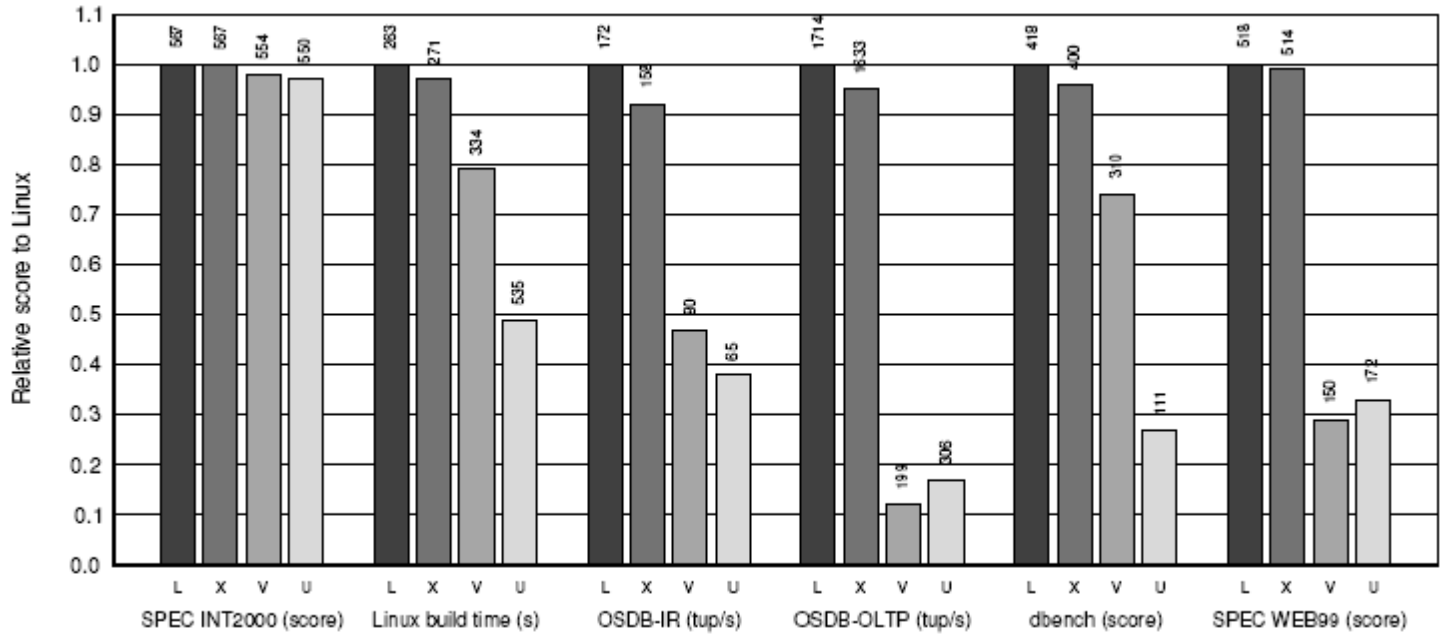


Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

XEN : Operating System Benchmarks

Config	null call	null I/O	stat	open close	slct TCP	sig 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	2p 0K	2p 16K	2p 64K	8p 16K	8p 64K	16p 16K	16p 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

XEN : Operating System Benchmarks

- Table 5, mmap latency and page fault latency.
  - Despite two transitions into Xen per page, the overhead is relatively modest.
- Table 6, TCP performance over Gigabit Ethernet LAN.
  - Socket size of 128kb
  - Results are median of 9 experiments transferring 400MB
  - Default Ethernet MTU of 1500 bytes and dial-up MTU of 500-byte.
  - XenoLinux's page-flipping technique achieves very low overhead.

Config	0K File		10K File		Mmap	Prot	Page
	create	delete	create	delete	lat	fault	fault
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

	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

XEN : Concurrent Virtual Machines

- In Figure 4,Xen's interrupt load balancer identifies the idle CPU and diverts all interrupt processing to it, and also the number of domains increases, Xen's performance improves.
- In Figure 5, Increase in number of domains further causes reduction in throughput which can be attributed to increased context switching and disk head movement.

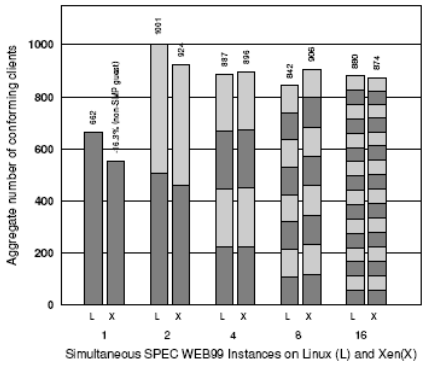


Figure 4: SPEC WEB99 for 1, 2, 4, 8 and 16 concurrent Apache servers: higher values are better.

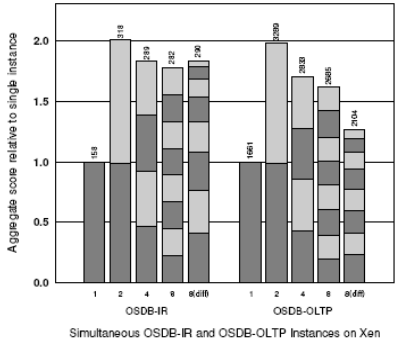


Figure 5: Performance of multiple instances of PostgreSQL running OSDB in separate Xen domains. 8(diff) bars show performance variation with different scheduler weights.

XEN : Scalability

- They examine Xen to sc
  - The minimum physical me
    with XenoLinux is 64MB.
    20kB of state per doma
  - Figure 6, performance c
    between large number c

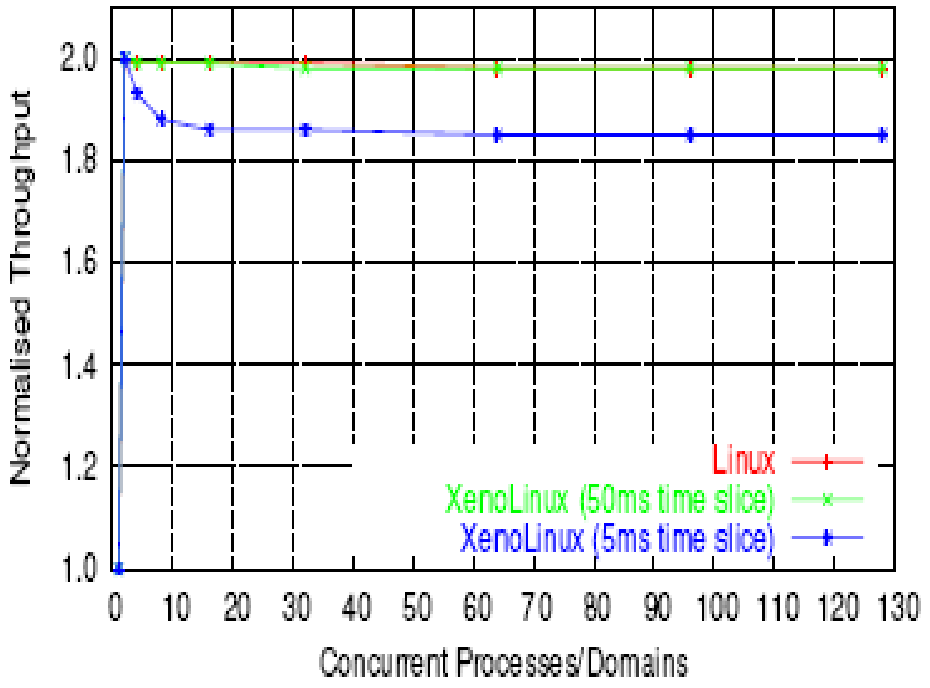


Figure 6: Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains

Debate

- What should a VMM actually "do"?
  - Hand: Argues that Xen is the most elegant solution and that the key is to efficiently share resources while avoiding "trust inversions"
  - Disco: Premise is that guest O/S can't easily be changed and hence must be transparently ported
  - Heiser: For him, key is that smaller kernel can be verified more completely (leads to L4... then SEL4)
  - Tornado, Barrelfish: Focus on multicore leads to radically new architectures. How does this impact virtualization debate?