VIRTUALIZATION

Xen and the Art of Virtualization

Are Virtual Machine Monitors Microkernels Done Right?

Presented by Brett Fernandes

Problems with Other Architectures

- Microkernels
  - Poor Performance
    overhead from IPC
  - Change the ABI
    Must forfeit all available software for the system
  - Monolithic kernel in disguise?
    Failure conditions of external pagers
- Exokernels
  - No application multiplexing
- No place for the untrustworthy!

Virtual Machines to the rescue?

- Excellent Performance
  - Achieved through Paravirtualization
- Retain the same ABIs
  - All required architectural features are virtualized
- Internal Paging by each VM
- Application multiplexing is everything
  - Each guest OS can multiplex applications securely
- The untrustworthy are welcome
  - Strong resource isolation between VMs

VMs - The resurgence rather than the emergence

- An old idea - IBM 370 in 1972.
  - A Virtual Machine Time-Sharing System (Meyer and Seawright) described the CP-67/CMS – the first virtual machine.

- Newer ventures:
  1. Vmware ESX Server (2001) - successor of Disco
  2. The Denali project (2001) - coined the term paravirtualization
  3. Sun's VirtualBox (2008)
  4. Microsoft released Hyper-V (2008)
  5. Xen is the most widely used by far – available as open source but now owned by Citrix Inc.

Xen and the Art of Virtualization

- **Paul Barham**

  Microsoft Research, UK

  Nemesis OS (QoS for I/O and virtual memory)
- **Rolf Neugebauer**

  Intel Research, Cambridge, UK
- **Boris Dragovic**

  XenoServer Team (Cambridge 2002), LinSec – Linux Security System
- **Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Ian Pratt**

  Cambridge University, UK
- **Andrew Warfield**

  University of British Columbia

Introduction

- Challenges to build virtual machines
  - Performance isolation
    - Scheduling priority
    - Memory demand
    - Network traffic
    - Disk accesses
  - Support for various OS platforms
  - Small performance overhead

Xen Principles

- Unmodified Application Binaries
  - No change to applications required
- Full multi-application OS support
  - Support for XenoLinux and ongoing work on Windows XP and BSD
- Paravirtualization
  - High performance
- Resource Isolation
  - Allows malicious users without harming other VMs
- Partial view of physical resources provided

Xen: Approach and Overview

- Multiplexes resources at the granularity of an entire OS
  - As opposed to process-level multiplexing
  - Price: higher overhead

- Target: 100 virtual OSs per machine
  - Denali supported over a thousand

Xen: Approach and Overview

- Conventional approach - Full virtualization
  - Virtual hardware is functionally identical to underlying machine
    - Virtualizing the entire instruction set
      - No view of physical resources
    - Problematic for certain privileged instructions
      - Failed silently rather than trapping
    - Shadow structures
      - Vmware traps every update page table event
    - No real time available
    - Hosted OS not modified

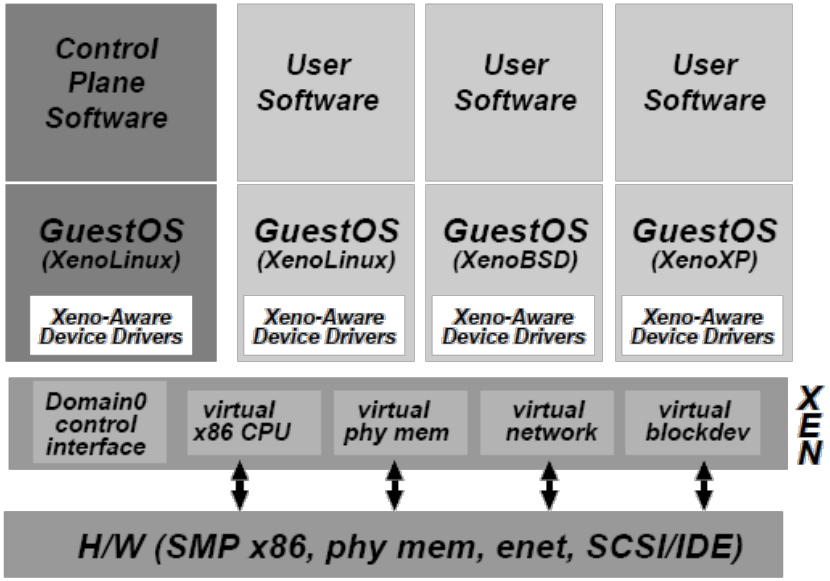
Xen: Approach and Overview

- New approach - paravirtualization
  - Virtual hardware is similar, not identical to the underlying hardware

    - Partial view of the underlying hardware
    - No modification of applications
    - VMs handle paging
      - No shadow tables required
    - Real, virtual and clock time provided
    - Need modifications to the OS
      - porting to Xen for every version of every OS

System Control Mechanism

- Separation of policy and mechanism
- Domain0 hosts the application-level management software
  - Creation and deletion of virtual network interfaces and block devices



System Control Mechanism

- Control Transfer: Hypercalls and Events

- Hypercall: synchronous calls from a domain to Xen
  - Analogous to system calls
- Events: asynchronous notifications from Xen to domains
  - Replace device interrupts

CPU Design

- X86 supports 4 levels of privileges
  - 0 for OS, and 3 for applications
  - Xen downgrades the privilege of OSes
  - System-call and page-fault handlers registered to Xen
  - “fast handlers” for most exceptions, Xen isn’t involved

CPU Implementation

- Borrowed virtual time scheduling
  - Allows temporary violations of fair sharing to favor recently-woken domains
  - Goal: reduce wake-up latency

Time and Timers

- Xen provides each guest OS with
  - Real time (since machine boot)
  - Virtual time (time spent for execution)
  - Wall-clock time
- Each guest OS can program a pair of alarm timers
  - Real time
  - Virtual time

Memory Design

- The conventional easier approach:
  - Software managed TLB
    - Associate address space IDs with TLB tags
    - Allow coexistence of OSes
    - Avoid TLB flushing across OS boundaries

Memory Design

- X86 does not have software managed TLB
  - Xen exists at the top 64MB of every address space
  - Avoid TLB flushing when an guest OS enters/exits Xen
  - Each OS can only map to memory it owns
  - Writes are validated by Xen

Physical Memory Implementation

- Reserved at domain creation times
- Memory statically partitioned among domains
- XenoLinux's balloon driver
- Does not guarantee contiguous allocation of memory

Virtual Address Translation

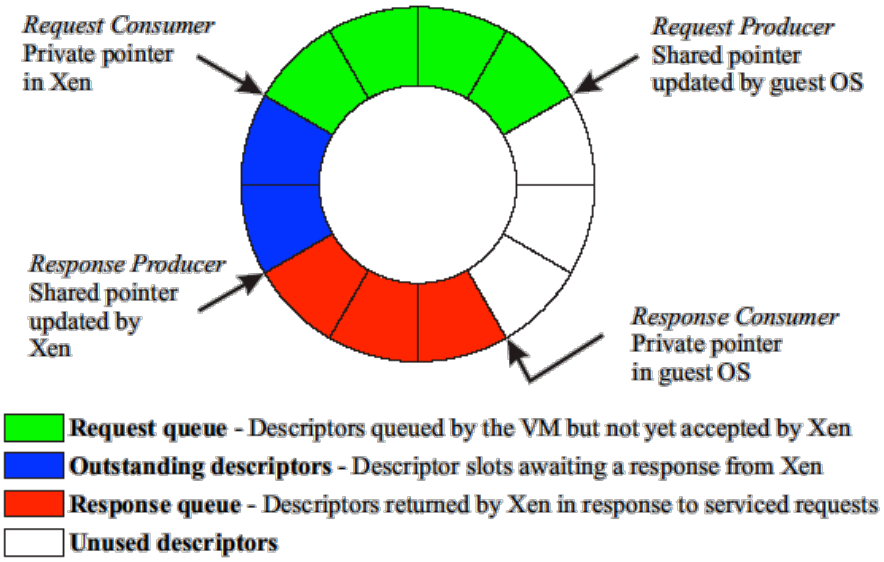
- No shadow pages (VMWare)
- Xen provides constrained but direct MMU updates
- All guest OSes have read-only accesses to page tables
- Updates are batched into a single hypercall

Device I/O Design

- Xen exposes a set of simple device abstractions
- Allows an efficient interface which provides protection and isolation
- I/O data transfer between domains via Xen

Data Transfer: I/O Rings

- Zero-copy semantics



Disk Access Implementation

- Only Domain0 has direct access to disks
- Other domains need to use virtual block devices
  - Use the I/O ring
  - Reorder requests prior to enqueuing them on the ring
  - If permitted, Xen will also reorder requests to improve performance
- Use DMA (zero copy)

Network

- Virtual firewall-router attached to all domains
- Round-robin packet scheduler
- To send a packet, enqueue a buffer descriptor into the transmit ring
- Use scatter-gather DMA (no packet copying)
  - A domain needs to exchange page frame to avoid copying
  - Page-aligned buffering

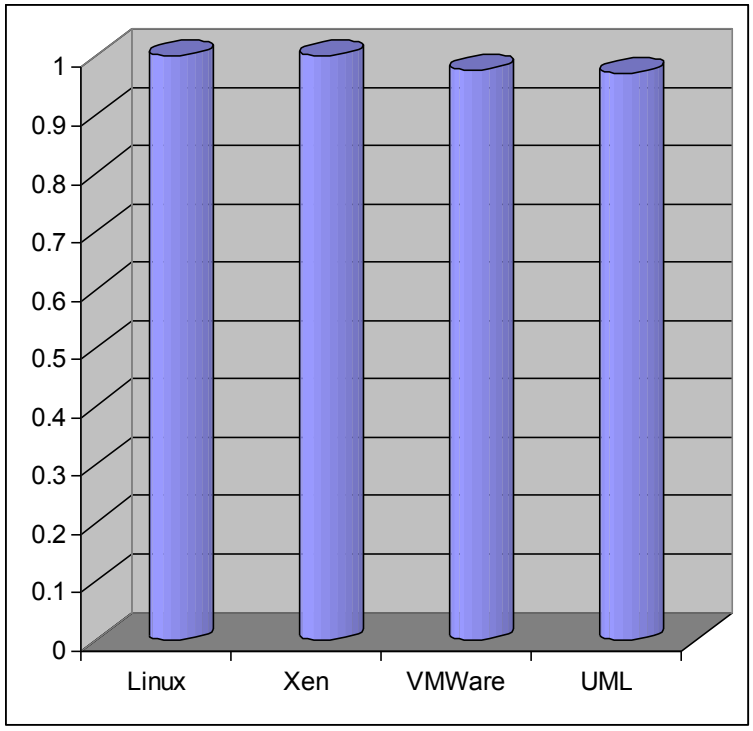
The Cost of Porting an OS to Xen

- Architecture Independent (78 lines)
- Virtual Block Device driver (1070 lines)
- Virtual Network driver (484 lines)
- Xen specific (1363 lines)
- < 2% of code-base

Evaluation

- Against other virtualization techniques
  - Vmware, User Mode Linux(UML)
- Single Native OS vs Virtual Machine
  - Running multiple applications on a native OS vs a guest OS
- Performance Isolation between Guest OSs
- Overhead of running large number of OSs

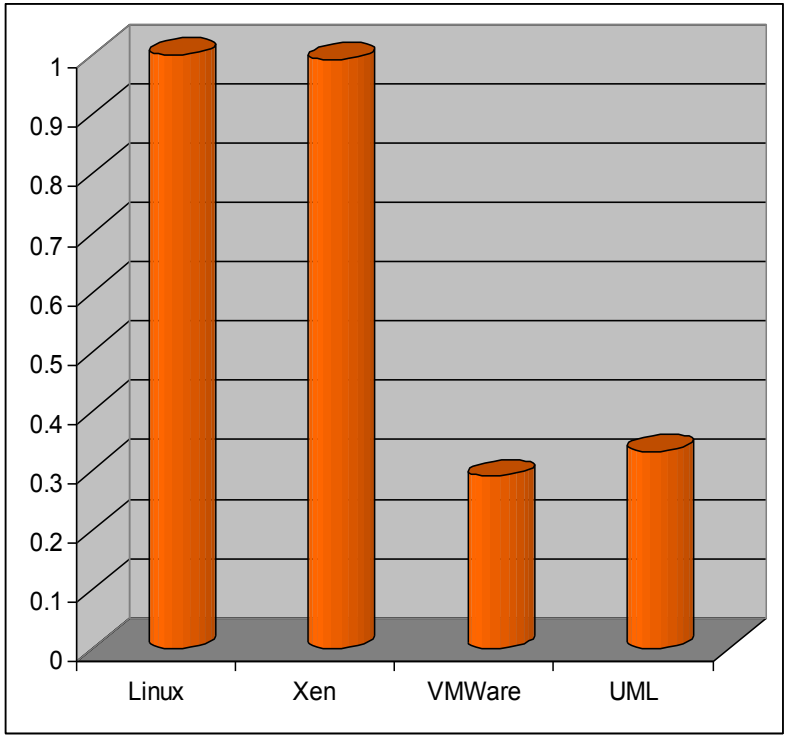
Relative Performance



SPEC INT2000 score

CPU Intensive

Little I/O and OS interaction



SPEC WEB99

180Mb/s TCP traffic

Disk read-write on 2GB dataset

O.S Benchmarks

Context switching times – extra overhead due to hypercall required to change the page table base.

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

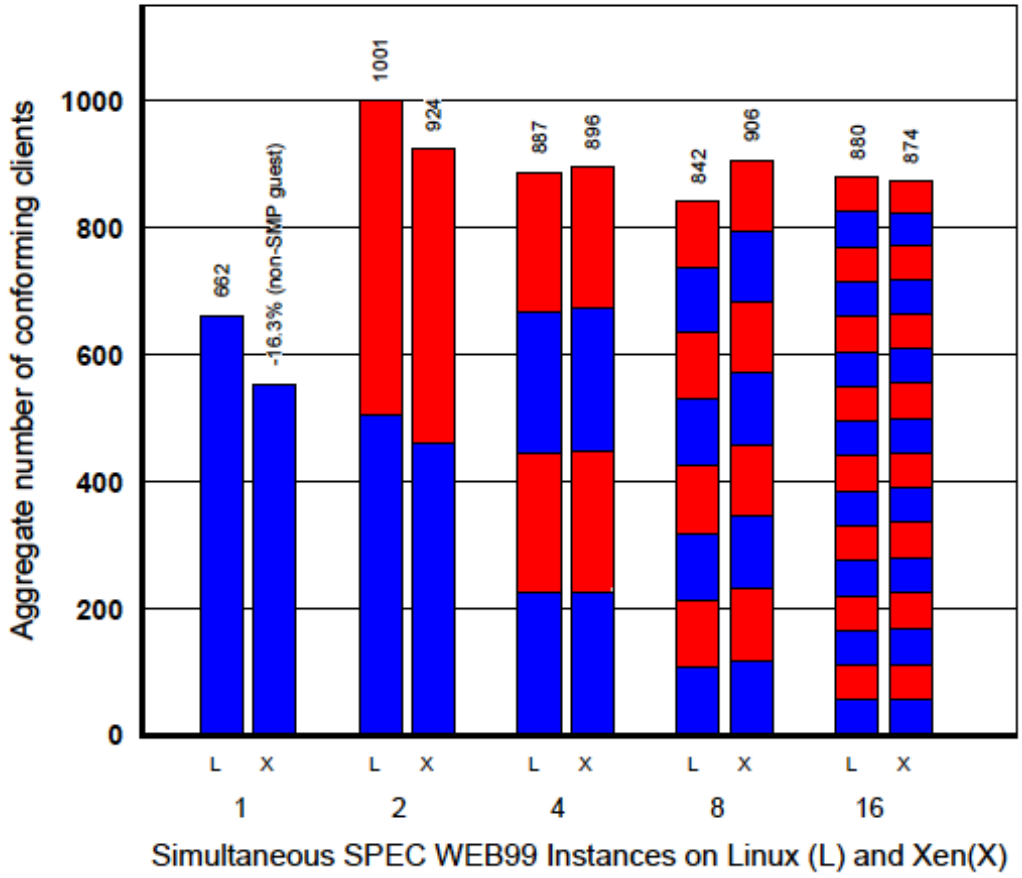
Table 4: `lmbench`: Context switching times in μs

Concurrent Virtual Machines

Multiple Apache processes in Linux

vs.

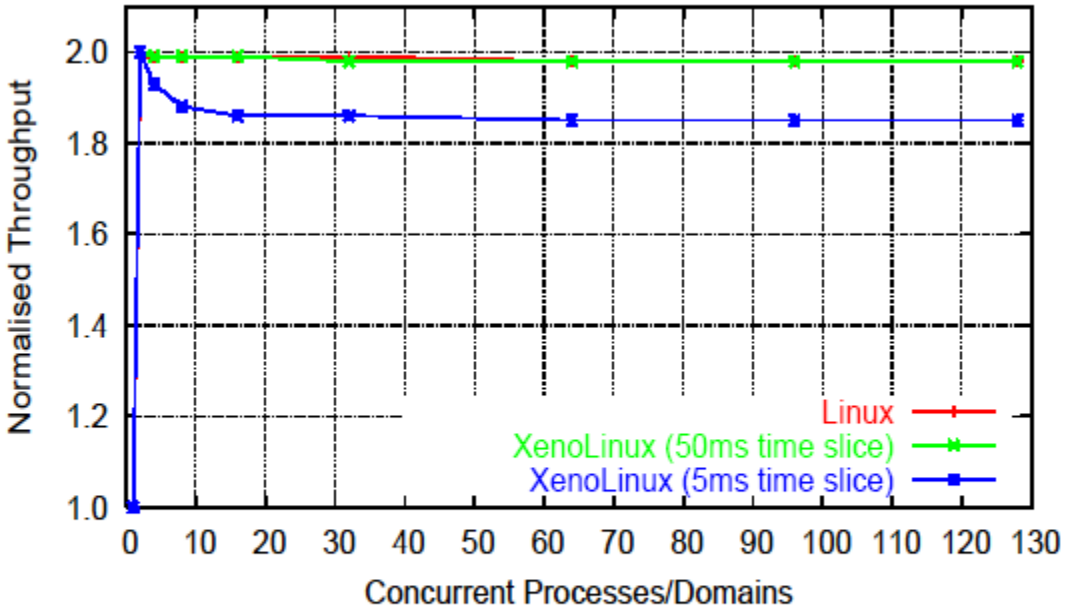
One Apache process in each guest OS



Performance Isolation

- 4 Domains
- 2 running benchmarks
- 1 running dd
- 1 running a fork bomb in the background
- 2 antisocial domains contributed only 4% performance degradation

Scalability



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

Issues

- Extra effort is required to port every version of every OS to Xen
  - Demonstrated by the 'ongoing effort' to port Windows XP and BSD
- Running a full OS is more taxing in terms of resource consumption
- The requirement of every privileged instruction being validated by Xen results in performance overhead
- Difficult to implement on an architecture with only 2 privilege levels

Discussion/Takeaways

- Main achievement – performance.
  - Completely outperformed Vmware in almost all benchmarks
- Identified potential problems and took steps to minimize them
  - Eg Fast exception handler for system calls
- OS level multiplexing
  - Solved the problem of performance isolation that plagued traditional OS techniques
- Innovative approach to TLB
  - Allocation of top 64MB to Xen avoids TLB flushes

Are Virtual Machine Monitor Microkernels Done Right?

- Steven Hand, Keir Fraser, Evangelos Kotsovinos
  Cambridge University, UK

- Andrew Warfield
  University of British Columbia

- Dan Magenheimer
  HP labs, Fort Collins
  Wrote the first PA-RISC simulator
  Developed Vblades, the first Itanium VMM

Sparking the Debate

- Mendel Rosenblum's claim
- VMMs are microkernels done right
- Common system goals
- Microkernels – Academia vs VMMs - Industry

Microkernels – Noble Idealism

- Communication oriented
- A smaller OS core is easier to maintain, validate and port
- Architecturally better than monolithic kernels

VMMs – Rough Pragmatism

- Strong resource isolation
- Main concern is reducing overhead due to extra layer
- Support execution of out-of-the-box applications
- Where do Exokernels stand?

Architectural Lessons

Liability inversion

External pagers in microkernels vs Parallax using VMs for storage

IPC Performance

Minimum communication between VMs

Decoupling of control and data path operations

OS as a Component

Microkernels forfeit the software available

VMMs appeal to developers because of a familiar environment

Discussion

- Very biased view of the debate
  - Possibly due to several of the authors working on Xen
- Focused on microkernel flaws and how VMMs were the answer
  - (almost certainly) Knowingly chose to refer to certain aspects of VMMs ambiguously
- Microkernels and VMMs appear to be more related rather than significantly different.