TCSS 422: OPERATING SYSTEMS

Virtualization

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

OBJECTIVES

- Virtualization
- Server consolidation
- VM hypervisors
- Virtualization overhead
- Virtual infrastructure management

VIRTUALIZATION MOTIVATIONS

- Server Consolidation
- Support legacy applications
  - Run old OSes, libraries, while masking new hardware changes
- Sandboxing / Isolation
  - Use VMs to isolate parts of applications
  - VMs act as containers
- Simulate unavailable hardware
  - Example: smart phone application development

VIRTUALIZATION MOTIVATIONS - 2

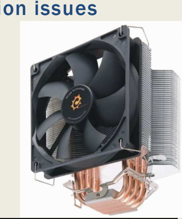
- Address datacenter underutilization
  - Run multiple different OSes simultaneously on same hardware
- Enable “application appliances”
  - Package application as set of virtual machines (or containers)
  - Encapsulate complex application configuration & setup
- Support Server Partitioning
  - Distribute server resources (e.g. RAM, CPU cores) across set of VMs
- Support software testing
  - Scalable tests, debugging at the OS/VM level

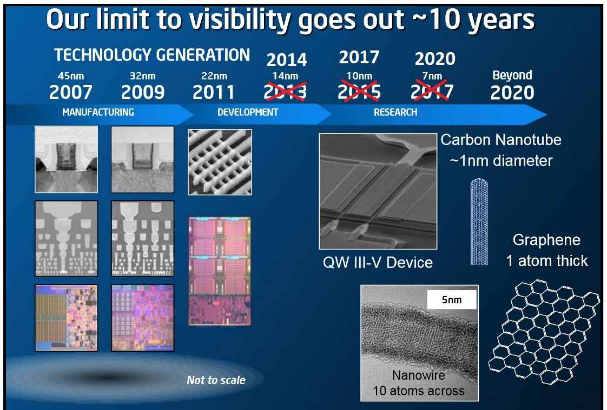
HISTORY OF VIRTUALIZATION

- Began with mainframe computers
- IBM System/370 circa 1972
- Original “hypervisor”
  - Control program
  - “Computing environment” for users to interact w/ mainframes
  - Time sharing(CPU), task isolation
  - Implements a “Virtual Machine” complete with unique memory address space

MOORE'S LAW

- Number of transistors in microprocessors doubles every two years
- Transistor density → heat dissipation issues
- Addressed by:
  - Smaller (shrink) die sizes
  - Lowering chip voltages
  - Aggressive cooling
  - Clock frequencies steady or no longer increasing





MOORE'S LAW: APPROACHING PHYSICAL LIMITS

- 1989, 800 nm, 5v, 1 core
- 1995, 350 nm, 2.8v, 1 core
- 1998, 250 nm, 2.0v, 1 core
- 1999, 180 nm, 1.6v, 1 core
- 2002, 90 nm, 1.2 - 1.4v, up to 2 cores
- 2006, 65 nm, 1.1125v, up to 4 cores
- 2008, 45 nm, 1.05 - 1.15v, up to 8 cores
- 2010, 32nm, .725 - 1.4v, up to 10 cores
- 2012, 22nm, .65 – 1.3v, up to 18 cores
- 2014, 14nm, .55 – 1.52v up to 24 cores
- 2017, 10nm …

MULTI-CORE PROCESSORS

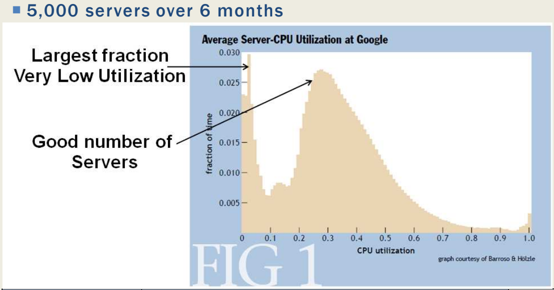
- Performance gains no longer achieved by increasing clock rate
- Performance gains by increasing number of CPU cores & architecture improvements
- Transistors are cheap
- Intel Xeon Westmere-EX
- 2.6 billion transistors, 10-cores

MODERN SERVERS

- X86 Multi-CPU Many-Core Servers
- Many servers 2 to 4 CPU sockets
- CPUs feature 8,10,12,16+ cores!
- CPU Hyper-threading (cores w/ 2 threads of execution)
- Most existing software not highly parallel
- 8-core CPU running 1-core code is no faster
- Parallel programming inherently difficult

AVERAGE SERVER UTILIZATION @ GOOGLE

- 5,000 servers over 6 months



WHAT ARE WE GOING TO DO WITH ALL OF THESE CORES?

- Use VMs to consolidate legacy physical servers
- For better utilization of physical resources
  - Parallel computing with "parallel" computers
  - Ideal for service-based computing
  - Process multiple simultaneous sessions
- Virtual machines share CPU, disk, memory
- Enables utility "Cloud" computing
  - provide VMs as a service (Amazon EC2)

SERVER CONSOLIDATION

SERVER CONSOLIDATION - 1

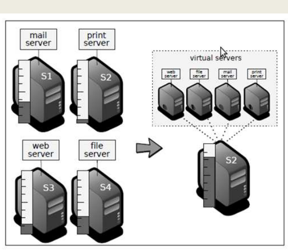
- Breaks traditional 1:1 mappings of
- Application to Operating System
- Operating System to Physical Machine
- Now n:1 mapping of
- (n) applications to (1) Physical Machine
- And 1:n
- (1) Application, (n) many physical machines

SERVER CONSOLIDATION STEPS

1. Characterize Application Resource Requirements
   - How sensitive is app to resource shortages?
   - Test in isolated environment to quantify resource requirements
2. Determine VMs distribution across Physical Infrastructure
3. Balance server workloads at runtime

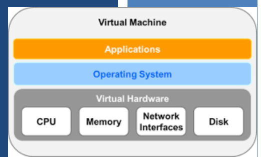
SERVER CONSOLIDATION - 2

- Computational resources of average physical server exceeds needs of applications
- Average server utilization only ~15%
- Virtualization enables server consolidation using virtual machines



SERVER CONSOLIDATION - 3

- Static allocation
- Direct replacement of physical servers with long running virtual machines
- Provides "base" level resources for application
- Dynamic allocation
- VMs added to meet application demand
- Shorter lifetime



VIRTUAL MACHINE HYPERVISORS

KEY VIRTUALIZATION TRADEOFF

- What is the tradeoff space?

Hardware Abstraction

Overhead

TYPE 1 HYPERVISOR

- Acts as a control program
- Miniature OS that manages VMs
- Runs on bare metal
- Also known as Virtual Machine Monitor (VMM)
- Traps instructions (i.e. device I/O) to implement sharing & multiplexing
- User mode instructions run directly on the CPU
- Paravirtualization
- Requires support to be included in the OS kernel
- **Objective: minimize virtualization overhead**

TYPE 2 HYPERVISOR

- Problem: Original x86 CPUs could not trap special instructions
- Instructions not specially marked
- Solution: Full Virtualization
- Trap ALL instructions
- "Fully" simulate entire computer
- Tradeoff: High Overhead
- Benefit: Can virtualize any operating system without modification

COMMON VMMS: FULL VIRTUALIZATION

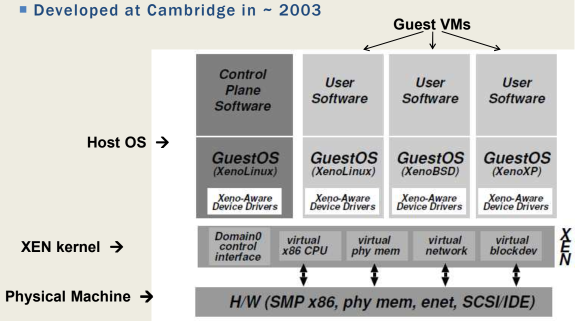
- Virtual Box
  - Commonly used for end-user MS Windows emulation
- Linux-based KVM
- XEN in hvm-mode

COMMON VMMS: PARAVIRTUALIZATION

- XEN
- Citrix Xen-server (a commercial version of XEN)
- VMWare ESX (commercial)
- VMWare ESXi (free)
- Paravirtual I/O drivers introduced
  - KVM
  - Virtualbox

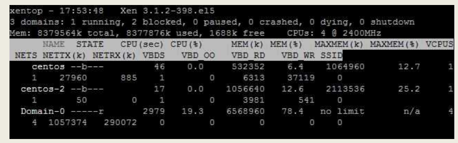
XEN

- Developed at Cambridge in ~ 2003



XEN - 2

- VMs managed as "domains"
- Domain 0 is the hypervisor (host OS)
- Domains 1..n are VMs



XEN - 3

- Physical machine boots special XEN kernel
- Kernel provides paravirtual API to manage CPU & device multiplexing
- Guests require modified XEN-aware kernels
- Xen supports full-virtualization for unmodified OS guests in hvm mode
- Amazon EC2 largely based on XEN

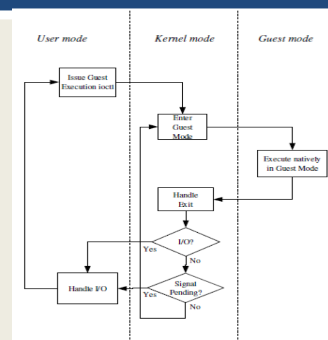
QEMU HYPERVISOR

- Generic process and machine emulator
- Written in 2005 by Fabrice Bellard
- Provides hardware emulation – full virtualization
- Emulates a variety of CPU architectures, on a variety of hosts:
  - X86, PowerPC server, PowerPC embedded, IBM S390
- Basis for KVM, VirtualBox, XEN-hvm mode
  - These are forked versions of QEMU codebase

KERNEL BASED VIRTUAL MACHINES (KVM)

- x86 hw notoriously difficult to virtualize
- Extensions added to 64-bit Intel/AMD CPUs
  - Provides hardware assisted virtualization
  - New "guest" operating mode
  - Hardware state switch
  - Exit reason reporting
  - Intel/AMD implementations different
    - Linux uses vendor specific kernel modules

KVM – 2



KVM – 3

- KVM has /dev/kvm device file node
  - Linux character device, with operations:
    - Create new VM
    - Allocate memory to VM
    - Read/write virtual CPU registers
    - Inject interrupts into vCPUs
    - Running vCPUs
- VMs run as Linux processes
  - Scheduled by host OS
  - Can be pinned to specific cores with "taskset"

KVM PARAVIRTUALIZED I/O

- KVM – Virtio
  - Custom Linux based paravirtual device drivers
  - Supersedes QEMU hardware emulation (full virt.)
  - Based on XEN paravirtualized I/O
  - Custom block device driver provides paravirtual device emulation
    - Virtual bus (memory ring buffer)
    - Requires hypercall facility
    - Direct access to memory

KVM DIFFERENCES FROM XEN

- KVM requires hardware-level VMX support
- KVM can virtualize any OS without special kernels
  - Less invasive
- Native KVM I/O performance is slow
  - Due to full hardware emulation

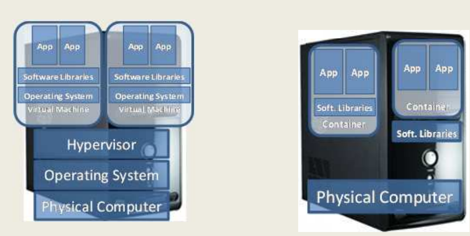
KVM ENHANCEMENTS

- Paravirtualized device drivers
  - Virtio
- Guest Symmetric Multiprocessor (SMP) support
  - Supported as of Linux 2.6.23
- Live Migration
- Linux Scheduler Integration
  - Optimize scheduler with knowledge that KVM processes are virtual machines

CONTAINER BASED VIRTUALIZATION

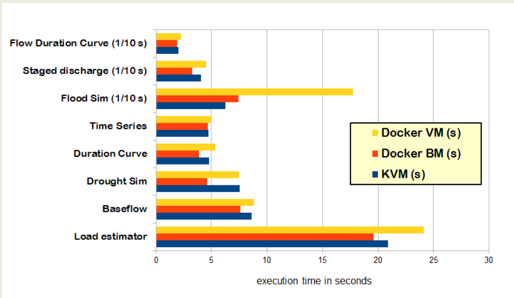
- VMs are soft partitions of the base OS
- All VMs share same OS kernel
- Tradeoff: No support for running different OSes
- Benefit: Faster & much less overhead
- Common containers:
  - Docker
  - CoreOS/Rocket
  - Linux-Vservers, OpenVZ *(legacy)*

CONTAINERIZATION



Virtualization **Containerization**

CONTAINERS VS VIRTUAL MACHINES



VIRTUALIZATION OVERHEAD

November 23, 2016

TCSS422: Operating Systems [Fall 2016]
Institute of Technology, University of Washington - Tacoma

L19.38

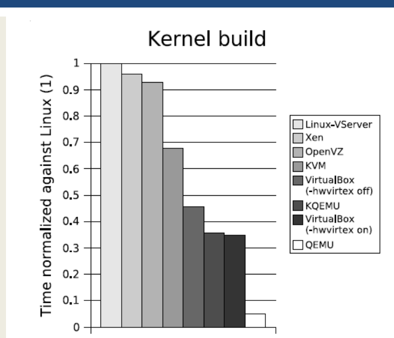
"VIRTUALIZATION OF LINUX SERVERS"

- Presented at 2008 Linux Symposium
- Benchmarks Virtualization Performance of:
  - Container-based virtualization
    - Linux-Vserver
    - OpenVZ
  - Paravirtualization
    - XEN
  - Full Virtualization
    - KVM
    - VirtualBox (/w VMX CPU support)
    - VirtualBox (w/o VMX CPU support)
    - KQEMU
    - QEMU

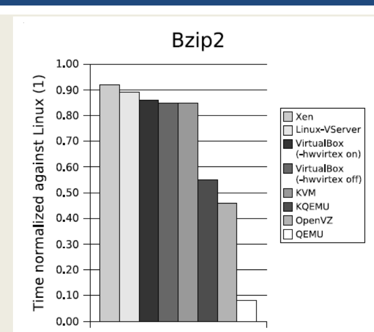
OVERHEAD TESTS

- Host OS: Ubuntu 7.10, 2.6.22-14 kernel
- Guest OS: Ubuntu 6.10 LTS
- IBM Lenovo desktop
  - Intel Core 2 Duo 6300 processor
  - 4GB RAM
  - 80GB SATA HD
  - 2x 1 Gigabit network interface cards
  - Single VM → 2GB RAM allocation
  - Multi VM: 2x1622MB, 4x811MB, 8x405MB, 16x202MB, 31x101MB

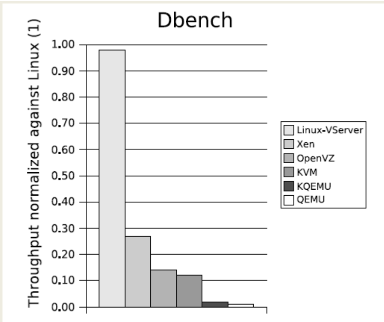
KERNEL BUILD (CPU & FILE I/O)



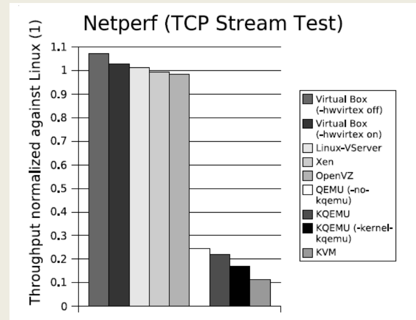
BZIP2 (CPU)



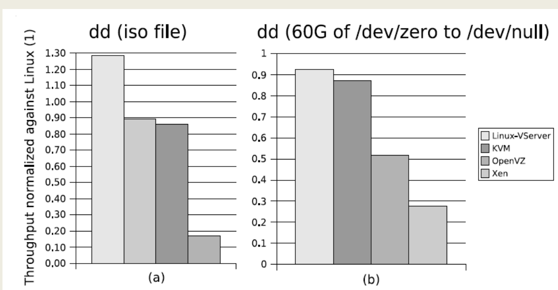
DBENCH (FILE I/O)



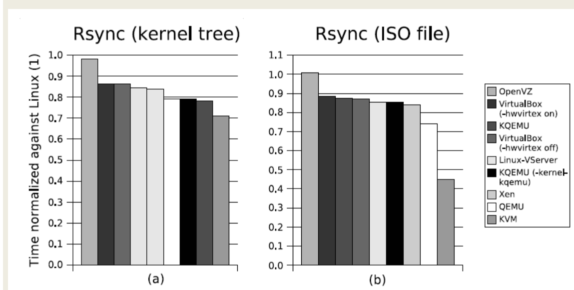
NETPERF (NETWORK I/O)



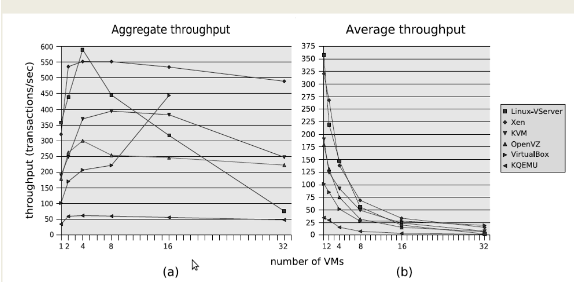
DD (FILE I/O)



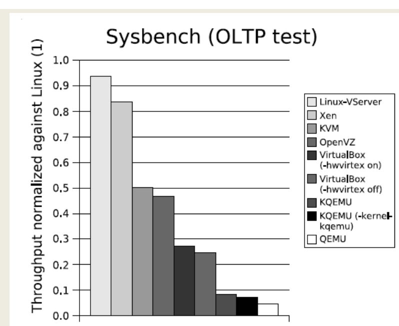
RSYNC (NETWORK I/O)



SYSBENCH (10,000 RDBMS TRANSACTIONS)



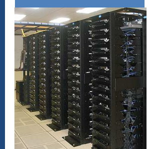
SYSBENCH (10,000 RDBMS TRANSACTIONS)



VIRTUALIZATION OVERHEAD – RUSLE2 EROSION MODEL

Hypervisor	Avg. Time (sec)	Performance
Physical server	15.65	100%
Xen 3.1	25.39	162.24%
Xen 3.4.3	23.35	149.20%
Xen 4.0.1	26.2	167.41%
Xen 4.1.1	27.04	172.78%
Xen 3.4.3 hvm	32.1	205.11%
KVM disk virtio	31.86	203.58%
KVM no virtio	32.39	206.96%
KVM net virtio	35.36	225.94%

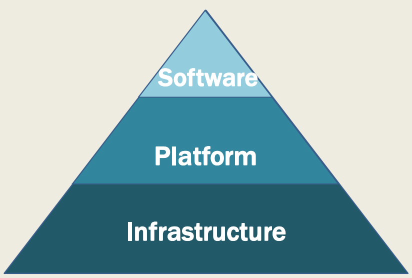
Average execution time for 100 model runs, 10 trials
15.8 sec CPU time, 56k dsr, 144k dsw, 9387k nbr, 9403k nbs
4 VMs hosted by 1 PM, 8 cores, ~4GB RAM,



VIRTUAL INFRASTRUCTURE MANAGEMENT

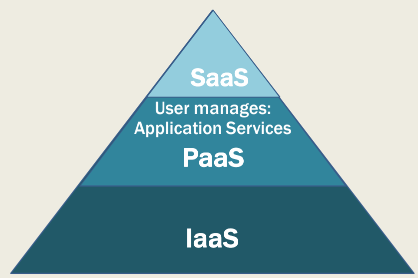
December 7, 2016 | TCSS422: Operating Systems [Fall 2016] Institute of Technology, University of Washington - Tacoma | L23.50

CLOUD COMPUTING – AS A SERVICE



August 7, 2012

CLOUD COMPUTING – AS A SERVICE



August 7, 2012

VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

- Middleware to manage virtual machines and infrastructure of IaaS "clouds"
- Examples
  - OpenNebula
  - Nimbus
  - Eucalyptus
  - OpenStack

VIM FEATURES

- Create/destroy VM Instances
  - Image repository
    - Create/Destroy/Update images
    - Image persistence
- Contextualization of VMs
  - Networking address assignment
    - DHCP / Static IPs
  - Manage SSH keys

VIM FEATURES - 2

- Virtual network configuration/management
  - Public/Private IP address assignment
  - Virtual firewall management
  - Configure/support isolated VLANs (private clusters)
- Support common virtual machine managers (VMMs)
  - XEN, KVM, VMware
  - Support via libvirt library

VIM FEATURES - 3

- Shared “Elastic” block storage
  - Facility to create/update/delete disk images
    - Amazon EBS
    - Eucalyptus SC
    - OpenStack Volume Controller

KEY/VALUE STORAGE

- Amazon Simple Storage Service (S3)
  - Used for object storage of arbitrary data
- Eucalyptus Walrus (S3 clone)
  - No replication
  - Hosted by single server
  - EC2 S3 compatible
- OpenStack -> ObjectStorage (S3 clone)
  - EC2 S3 compatible
  - Not used for VM images

VM IMAGE MANAGEMENT

- Image Repositories
  - Image registration/publication
    - Initial transfer from VM or physical host
    - Creation of repository copy
    - Replication across redundant servers
    - Performance
  - Snapshot live VMs
  - Metadata

VM IMAGE STORAGE

- Amazon -> S3
- Eucalyptus -> Walrus (S3 clone)
- OpenStack -> ImageService

EPHEMERAL STORAGE

- Hosted on physical machine with the VM
- Base image
  - Mounted as /dev/sda1
  - EC2 size limit = 10 GB
  - Limit applies to persisted portion
- Extended space
  - Larger non-persisted space
  - Mounted as /dev/sda2
  - Initially empty

"ELASTIC" BLOCK STORAGE

- Network storage service
  - Not co-located with VM
  - Amazon EBS
  - Eucalyptus SC
  - OpenStack Volume Controller
- Requires
  - Dedicated high speed server(s) with large disks
  - Network Attached Storage (NAS) device

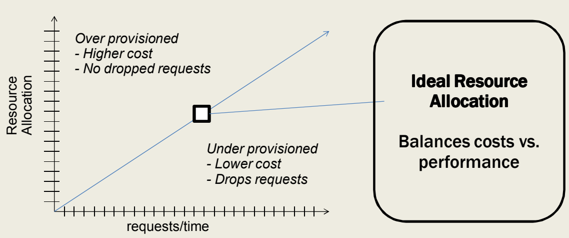
"ELASTIC" BLOCK STORAGE - 2

- Facilitates OS image separation from data
- Performance bounded by network
  - Amazon EBS - 1 Gigabit max
- VM Disk I/O becomes Network I/O + Disk I/O
- Amazon/Eucalyptus- Boot from EBS

SCALING INFRASTRUCTURE

- Multi-tier application scaling
  - Simply adding VMs may be insufficient
  - Which tier is the bottleneck?
    - Application server
    - Database server
    - Log/transaction server
    - Number of worker threads
    - Number of database connections

APPLICATION SCALING: PERFORMANCE VS. RESOURCES



QUESTIONS

December 7, 2016
TCSS422: Operating Systems [Fall 2016]
Institute of Technology, University of Washington - Tacoma
L23.66

EXTRA SLIDES

67

KVM I/O VIRTUALIZATION

- Programmed I/O (pio)
- Memory-mapped I/O (mmio)
- All pio and mmio requests forwarded to userspace
- Device model used to interpret requests
  - Simulates behavior
  - Triggers real I/O with underlying physical hardware as needed
  - Interrupts injected into guest when I/O complete

MEMORY MANAGEMENT UNIT (MMU) VIRTUALIZATION

- X86 provides virtual memory system with 1-level of mapping (page table)
  - Guest virtual -> guest physical
- Two-level mapping needed for hosting virtual machines
  - Guest virtual -> guest physical -> host physical

MMU - 2

- Classic Solution
  - CPU page table used as a "shadow"
    - Guest physical -> host physical
  - Guest (VM) page tables stored in memory
    - Above the "shadow" table
    - Enables combined translation
    - Guest virtual -> host physical
  - Guest (VM) page tables writes require synchronization with "shadow" page table

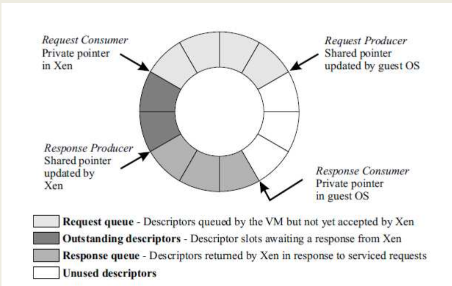
XEN MEMORY MANAGEMENT

- No virtual page table or address translation
- XEN provides all guests with direct read-only access to the memory management unit (MMU)
- Writes are validated by XEN by tracking types and reference counts
- Page table updates grouped into single hypercall

XEN PARAVIRTUAL I/O

- Uses Virtual Block Devices
- Physical devices shared by XEN using a circular queue
- Direct memory access used to transfer I/O directly to XEN VM memory
- Multiple requests batched together to improve throughput at the expense of latency
- Use of hypercalls enable VM to trigger privileged operations – (ring 0)

XEN DATA TRANSFER I/O RING



HYBRID VIRTUALIZATION

- Full virtualization -> faster CPU/memory mgmt
- Paravirtualization -> faster I/O
- Combine Full & Para for optimal performance
  - VMX CPU extensions
  - Paravirtualized device drivers
- Supported by XEN, KVM
  - XEN HVM
- Paravirtual I/O requires specialized drivers