Virtual Machines

Virtual Machine (VM)

Layered model of computation

Software and hardware divided into logical layers

Layer n

- Receives services from server layer n – 1
- Provides services to client layer n + 1

Layers interact through well-defined programming interface

Virtual layer

Software emulation of hardware or software layer n

Transparent to layer n + 1

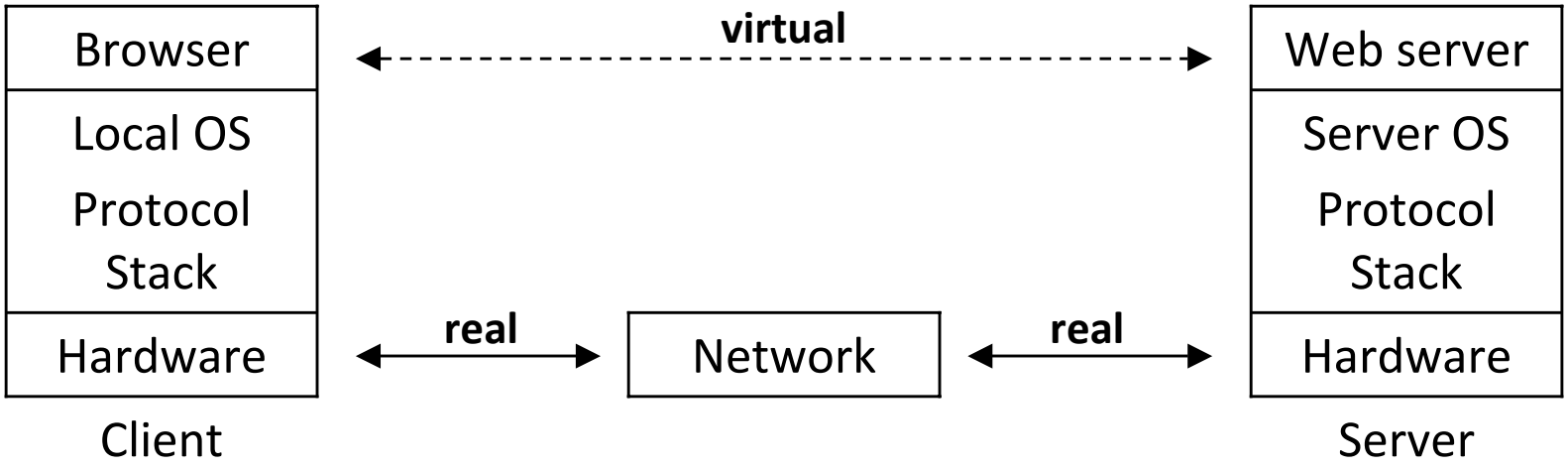
- Provides service to layer n + 1 as expected from real layer n

Virtual layer n can run at some layer m ≠ n in real system

Virtual System	Real System
n + 1	n + 1
n	virtual n = m
n – 1	m – 1

Examples of Virtual Systems

Web browser exchanges data with server



Cloud computing

- Virtual
  - Service level agreement (SLA) specifies infrastructure requirements
  - User sees hardware / software configuration / performance
- Real
  - Provider assembles virtual configuration
  - Meets SLA requirements
  - May be implemented in any way

Types of Virtual Machine

Process Virtual Machine

VM provides application interpretation above OS

Hosted Virtual Machine

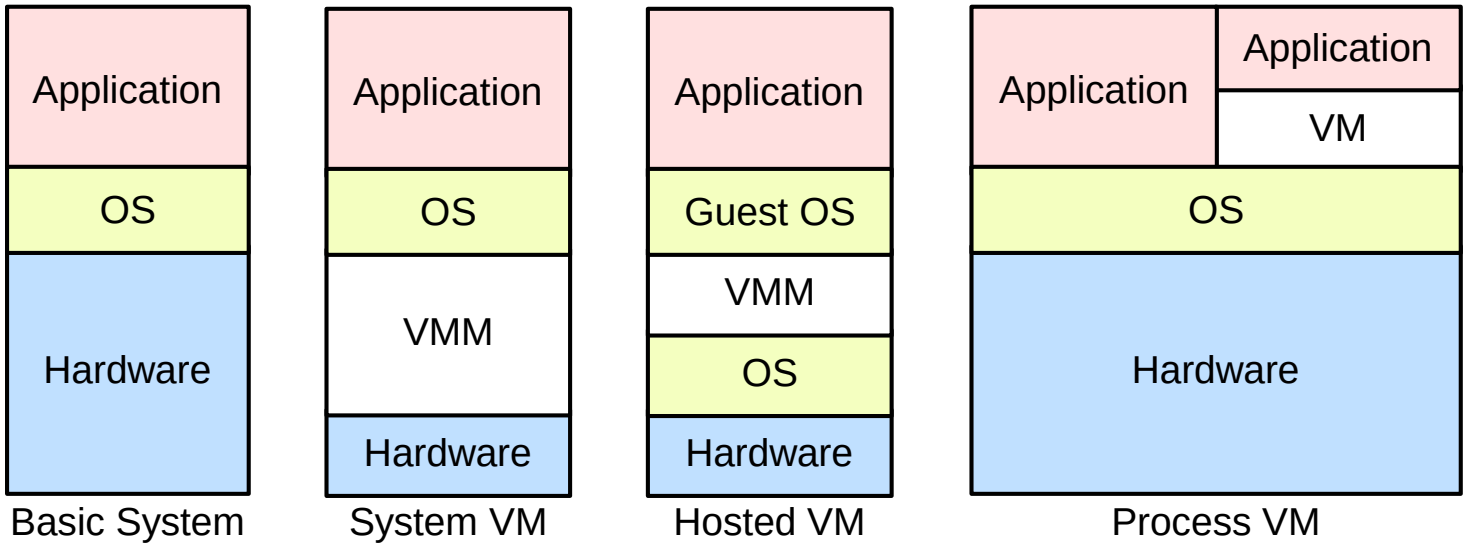
Virtual machine monitor (VMM)

- Runs above primary OS / below guest OS
- Provides guest OS with software emulation of real hardware system

System Virtual Machine

Emulation of system-level hardware environment

- Runs above physical hardware and below one or more OSs



Process VM Example — Java

Designed for program portability between platforms

Provides standard interface to software

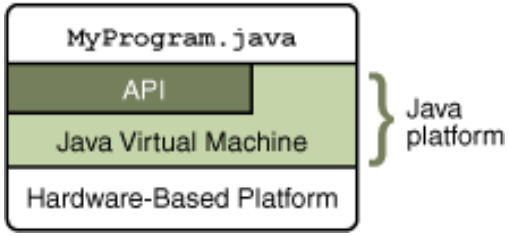
Java VM located above a standard OS

Interface to hardware implementation dependent

I/O operations performed by calls to OS

Java compiled to bytecode

Bytecode usually run (interpreted) in Java VM





Java without VM

Java bytecode processor in IBM mainframes

Native machine language (ISA) is Java bytecode

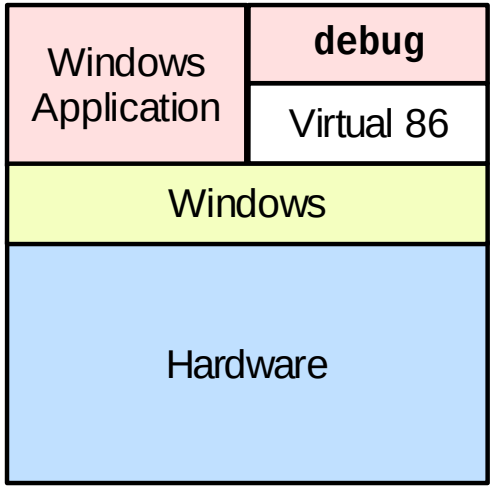
Execute Java bytecode without interpretation

http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

Hosted VM Example — Guest OS Over OS

DOS command line interface over Windows

- Windows allocates 1 MB virtual memory space
- Copies DOS kernel into low memory
  - System calls handled by guest DOS kernel
- DOS accesses to hardware
  - Trapped and served by Windows host OS
  - Responses returned to DOS
- Concurrent DOS windows
  - Multiple allocations of 1 MB virtual memory spaces
- Debug
  - Application running in virtual DOS machine
  - Sees 1 MB memory space allocated by Windows
  - Register values
    - Windows emulates real values to DOS
    - Debug emulates DOS values to user



Parallels, VirtualBox, VMware, DOSBox, ...

- Host Windows, Linux, DOS, ... as guest OSs over host OS

Virtual Machine in IBM z/990 Mainframe

Hardware

CPUs, I/O system, internal communication network

VMM (hypervisor)

Operator console for partitioning/configuring CPUs and I/O

Provides hardware emulation as abstraction to OS layer

OS

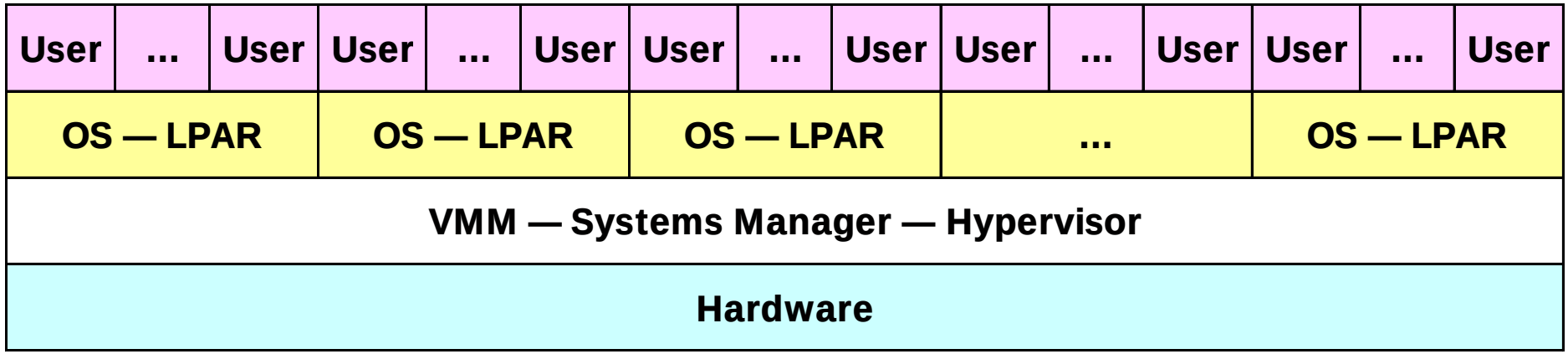
Logical partition (LPAR) runs separate instance of operating system

Run z/OS, MVS, VM, Unix, Linux, Windows, ... instances in parallel

Non-Windows OS versions expect to see hypervisor (not hardware)

User

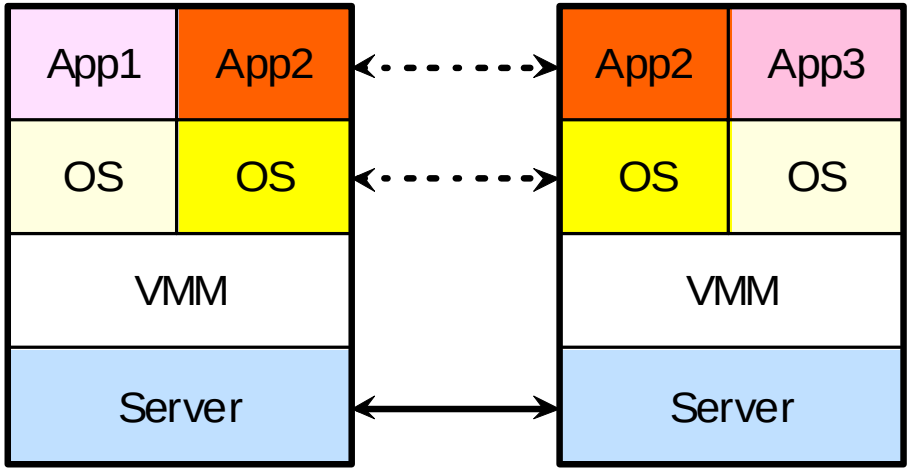
User sees single-user interface provided by one OS



VM as System Management Tool

Isolate user environments on single hardware platform

- Multiple copies of single operating system running independently
- Multiple operating systems running concurrently
- Maintain higher security



Resource management

- Hardware redundancy
  - High availability
  - Recovery management
- Hardware pooling
  - Assemble hardware cluster
  - Map applications to hardware efficiently
- Load balancing
  - Remap applications to hardware

Virtualization for Server Systems

Old file server model

- Run one application per physical server
- Server specified for worst case load
- Large number of typically underutilized servers
- Huge aggregate space capacity

Competition from mainframes

- VMM provides dynamic load balancing
- Hardware provides centralized power, cooling, monitoring, backup
- High SAR — scalability, availability, reliability
- Lower cost per served client than server farm

Virtualization in server

- Partition hardware resources to run independent applications
- Intel virtualization
  - IA-32 and IA-64 ISA support
  - I/O chipset support

HP Virtual Partitions (vPars)

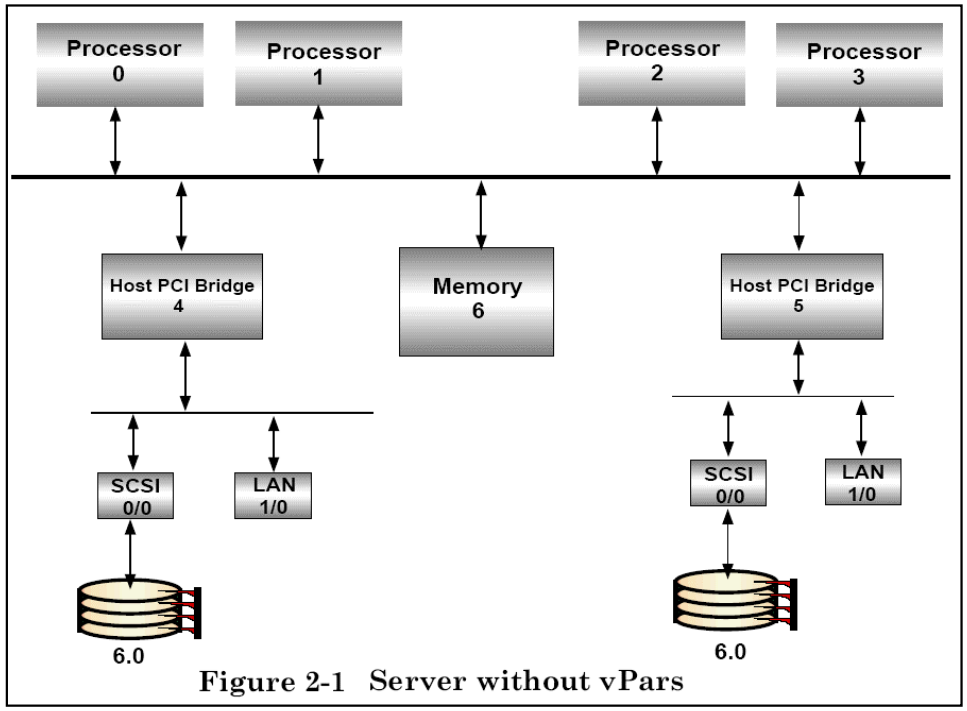


Figure 2-1 Server without vPars

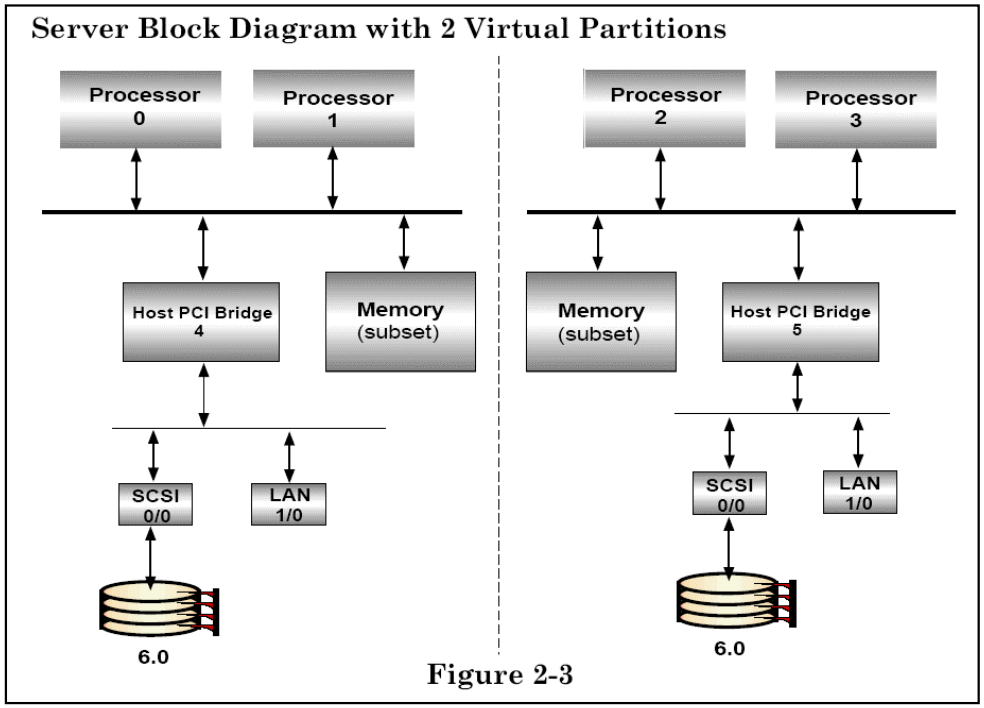
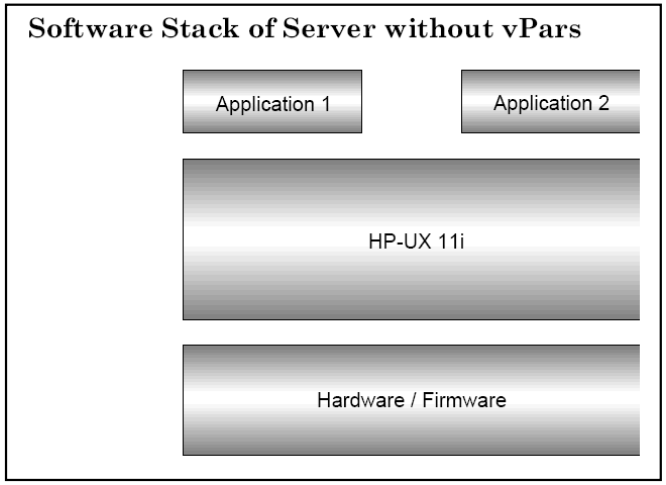
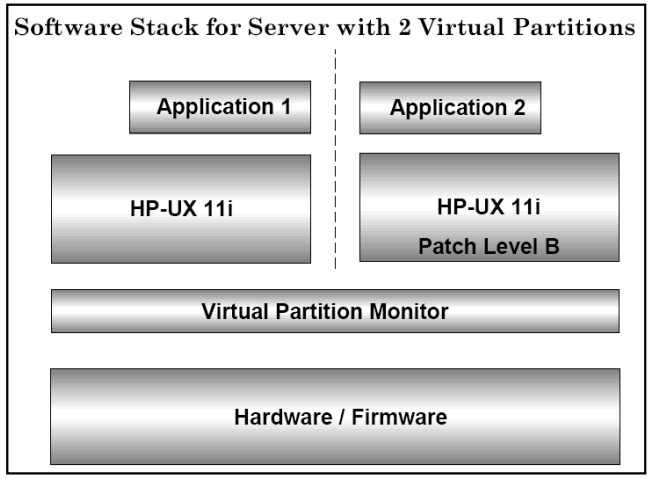


Figure 2-3



Boot Order



Hewlett-Packard, "Installing and Managing HP-UX Virtual Partitions (vPars)"

System VM Organization

Hypervisor

- Virtual machine monitor (VMM)
- Lowest layer above physical hardware (host)
  - Uniprocessor or multiprocessor system
- Creates virtual machine (VM) environments for OSs
  - Allocates physical host resources to virtual resources

VM overhead

- Processor intensive applications — low overhead
  - Infrequent use of OS calls
  - Most instructions run directly on hardware
- I/O intensive applications — high overhead
  - Frequent use of OS calls
  - OS calls for I/O services run in emulation
- I/O-limited applications
  - Program throughput limited by I/O latency
  - Emulation adds relatively small overhead

VMM Requirements

Hardware abstraction

Guest environment must replicate hardware

VMM must present well-defined emulated interface to OS

Protection

Isolate guests from one another

Protect VMM from guest OS and application software

Guest software cannot change allocation of physical resources

Privilege

VMM runs in kernel mode

Guest OSs and applications run in user mode

Hardware support for VMM

Virtualization primitives built into mainframe ISA

Any OS or application access to hardware causes trap to VMM

VMM catches every access to hardware abstraction layer (HAL)

Virtualization Awareness

Virtualization-unaware guest OS

- OS written to run above physical hardware
- Expects full and direct control of real hardware
- Requires extensive intervention and remapping by VMM

Virtualization-aware guest OS

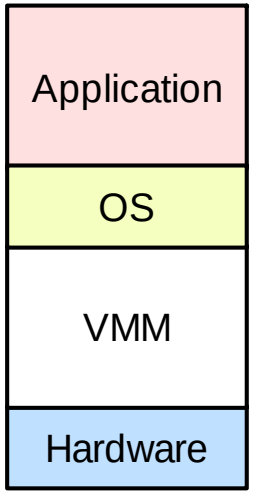
- OS written to run above VMM/hypervisor
- Expects to interact with virtual host
- Does not expect full or direct control of physical hardware
  - OS code interfaces with VMM code
  - No need to remap (bluff) pointers intended for real hardware
- May be presented with view of real system for limited operations
- Example — mainframe OS
  - Writes I/O outputs to hypervisor interface
  - Does not attempt to configure I/O hardware devices
  - Particular OS may be given direct control of particular I/O device

Hardware Emulation Activities

OS sees hardware through operations

OS instructions cause to CPU initiate memory and I/O operations

I/O devices initiate DMA operations and interrupts



Operation	Real Hardware	VMM Emulation
CPU Memory Access	Read data or instruction Write data	Translate data/instruction from guest to host format Remap address space Read/Write to real host memory
CPU I/O Device Access	Read data or instruction Write data	Translate data/instruction from guest to host format Remap I/O port space Read/Write to real host I/O device
I/O device actions	DMA or IRQ	VMM manages I/O device DMA Translate OS interrupt handlers from guest format to host format

Full/Partial System Emulation

Full system emulation

VMM intervenes in every OS access to hardware

CPU	**Translates guest ISA to host ISA**
Memory	**Translates memory size and organization**
Chipset	**Translates guest configuration instructions to host**
I/O devices	**Translates guest driver to host driver**

CPU emulation example

Run Nintendo game on PC

Translate each Nintendo instruction to IA-32 instruction set

Partial system emulation

Part of host hardware presented to OS unchanged

VMM passes guest operations to host with minimal intervention

Most system VMs emulate subset/superset of real host hardware

CPU emulation only in special cases

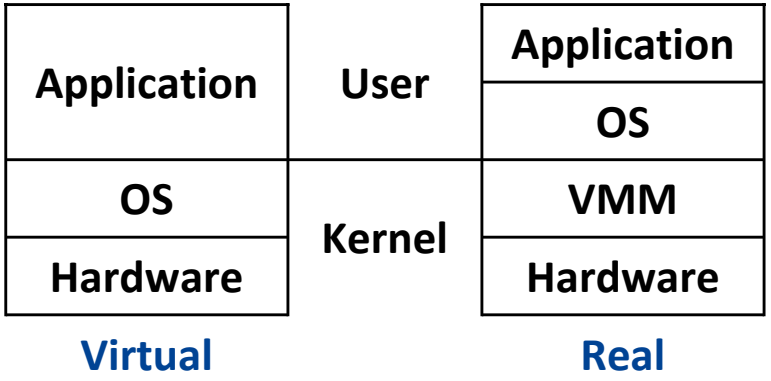
Virtualization Difficulties for IA-32

IA-32 designed to provide hardware support to OS

- Memory segmentation
- Virtual memory and paging
- Task management
- Interrupt management
- Protection and privilege for segmentation, paging, interrupts

Workaround virtualization

- Treat OS like user application
- Can create a kludge on IA-32 systems
- IA-32 operating systems
  - Expect to have highest privilege
    - Can easily discover their lower privilege
  - Expect to see full virtual memory space
    - Suffer memory and table compression
    - VMM must reserve address and table space for itself



Intel Virtualization Technology (VT)

Virtual machine monitor

- Hardware boots (3rd party) VMM software instead of OS
- VMM configures hardware resources among guest systems
  - Remaps hardware locations to virtual pointers for guests
  - OSs boot within guest partitions

Hardware support for virtualization

- VT-enabled processors alternate between operating modes
  - Root mode grants full hardware control to VMM
  - Non-root mode presents virtual pointers to guest OS
- VT-enabled chipset
  - Grants control of I/O to root mode
  - Remaps I/O channels for non-root mode

VMX non-root	User	Ring 3 User Privilege
	OS	Ring 0 Virtual Full Privilege
VMX root	VMM	Real Full Privilege

Operating system

- Sees virtual machine as real system
- Operates in ring 0 for maximum privilege
- Sends instructions to hardware pointers in usual way

http://www.intel.com/technology/itj/2006/v10i3/index.htm

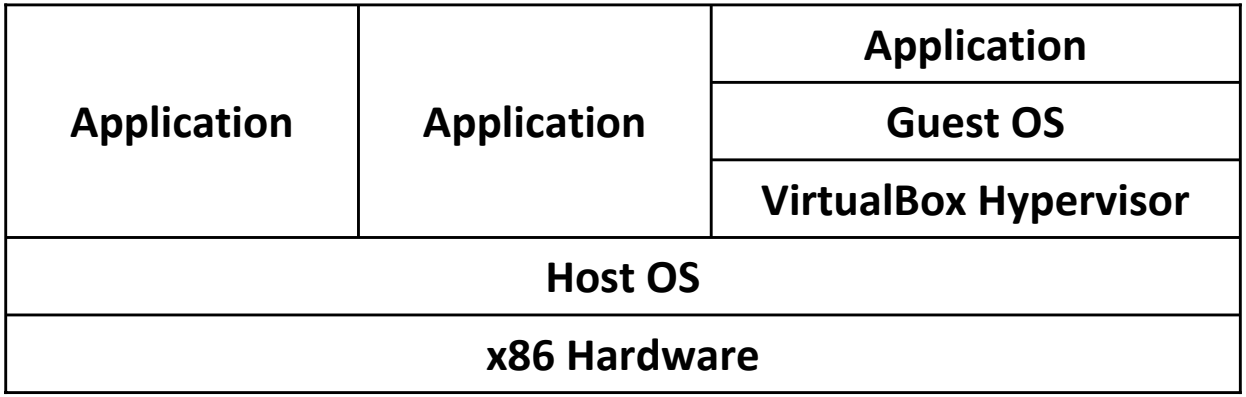
VirtualBox

Open source hosted VMM by Oracle (Sun Microsystems)

- Runs on Intel and AMD x86 hardware
- Runs above Windows, Linux, Mac OS X (Intel), Solaris
- Provides VM with guest OS
  - Standard DOS, Windows, Linux, OS/2, FreeBSD, Solaris
- Uses hardware virtualization support if available (not required)

Scheduling

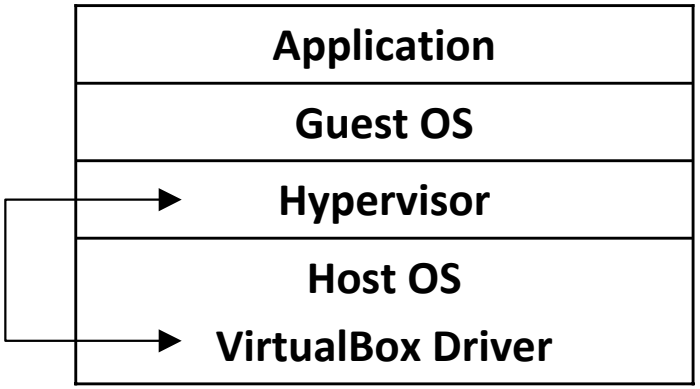
- Host OS grants timeslice to VM
- VM sub-processes scheduled by guest OS



VirtualBox Architecture

Front-end (client)

- VirtualBox hypervisor
- Runs above host OS
- Without Intel VT performs workaround virtualization
  - Hypervisor runs in ring 0 of guest context
  - Guest OS runs as user program in ring 1 of guest context
- Limited use of Intel VT if available



Back-end (server)

- Ring 0 driver in host OS
- Copes with "gory details of x86 architecture"
- Allocates physical memory for VM (guest OS)
- Saves/restores guest CPU context during host interrupt
  - Registers and descriptor tables
- No intervention in guest OS process management

CPU Operating Modes

Runs native (not emulated) on CPU

- Host applications at ring 3
- Host OS code at ring 0
- Guest "safe" application code at ring 3
  - Non-system activities
  - Makes system calls to guest OS

Runs emulated on CPU at ring 3

- Guest application code that causes guest OS interventions
  - Disable interrupts
  - Trap of prohibited accesses
  - Executes real mode code
- Each instruction interpreted by VirtualBox driver
  - CPU runs interpreted code instead of native code

Runs native on CPU at ring 1

- Guest OS ring 0 code
- VirtualBox driver handles "gory details" of workaround

Xen

Open source system VMM

- Runs on Intel and AMD x86 hardware
- Runs directly above hardware
  - Linux required to build and install Xen
- Provides VMs with guest OSs
  - Linux, Solaris, Windows XP, 2003 Server
- Hardware virtualization support required for Windows guest OS

Paravirtualization for Linux / Unix guest OS

- OS kernel modified to work with Xen explicitly
- Operating systems ported to run on Xen
  - Similar effort to porting OS to new hardware platform
  - Paravirtual machine architecture very similar to native hardware
- User space applications and libraries not modified

Xen Architecture Overview, http://wiki.xensource.com/xenwiki

Xen Architecture

Xen hypervisor

Directly above hardware

Boots system on start-up

Domain 0

Initialized by hypervisor on boot

Runs XenLinux — modified Linux kernel

Provides Domain Management and Control (DMC)

Domain U

VM running guest OS

<table>
<tr><td>DMC</td><td>Application</td><td>Application</td><td>Application</td></tr>
<tr><td>XenLinux OS
Domain 0</td><td>Guest OS
Domain U</td><td>Guest OS
Domain U</td><td>Guest OS
Domain U</td></tr>
<tr><td colspan="4">Xen Hypervisor</td></tr>
<tr><td colspan="4">x86 Hardware</td></tr>
</table>

Hypervisor

Full privilege

Operates directly on hardware in ring 0

Functions

CPU scheduling for virtual machines

Memory partitioning

Provides hardware abstraction to virtual machines

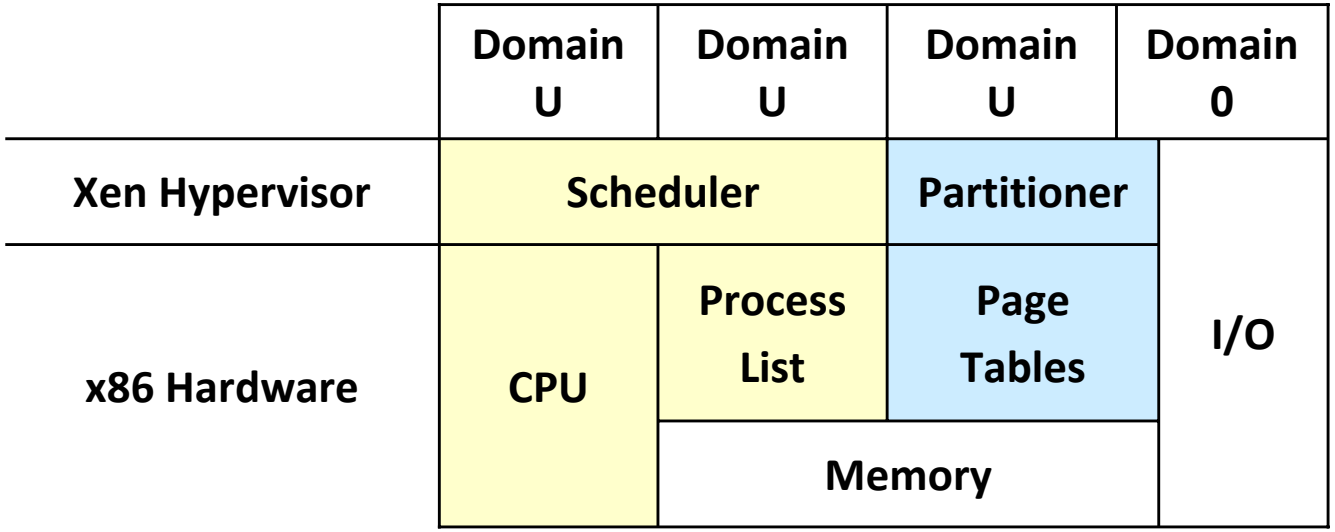
No awareness of

Networking

External storage devices

Video

Common I/O



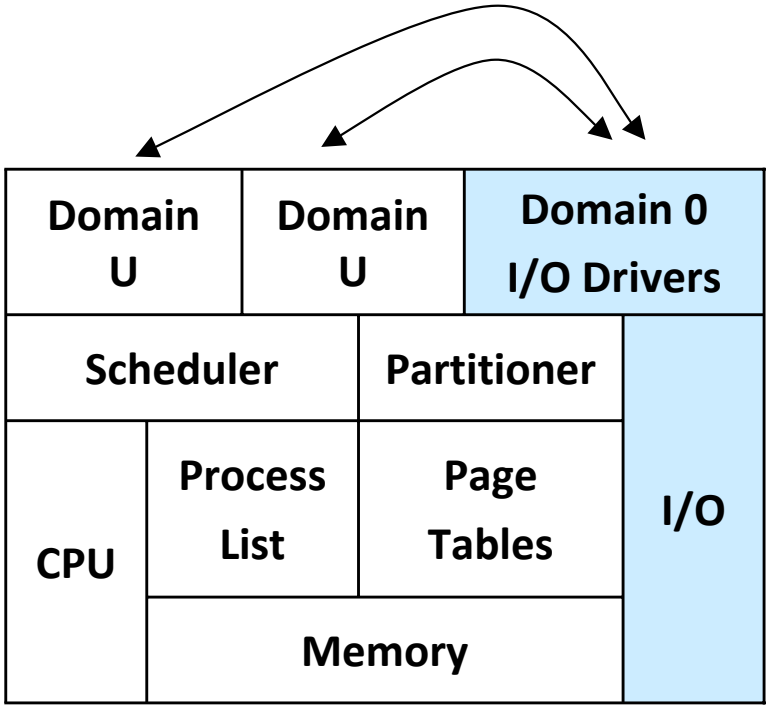
Domain 0

XenLinux

- Modified Linux kernel running in unique VM over hypervisor
- Direct privileged access rights to physical I/O resources
- Provides I/O virtualization to Domain U guest VMs

Generic I/O drivers

- Network Backend Driver
  - Manages local networking hardware
  - Processes all VM networking requests from Domain U guests
- Block Backend Driver
  - Manages local storage disk
  - Processes all read/write data requests from Domain U guests



Domain U Paravirtualization (PV)

Domain U PV guests

Paravirtualized VM running modified Linux/UNIX kernels

OS expectations

No direct access to host hardware

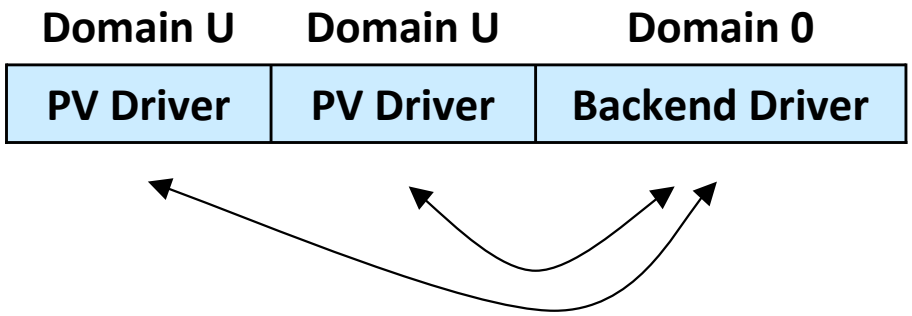
Shares host hardware with other VMs

Guest drivers provide I/O access

Access backend drivers in Domain 0

PV Network Driver

PV Block Driver



Domain U HVM

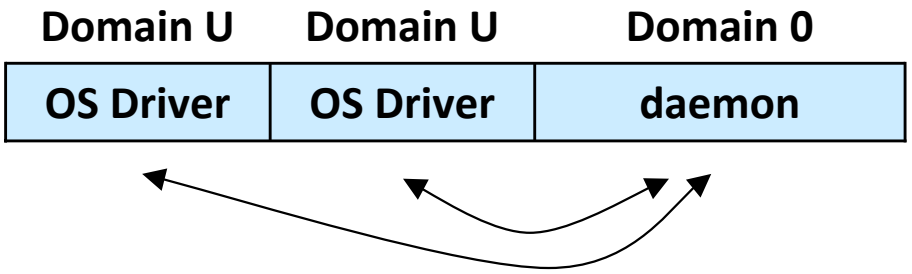
Domain U HVM Guests

Fully virtualized machines

Run standard Windows or other unmodified OS

OS runs as VMX non-root operation with VT-x

OS expectations



No hardware virtualization

Not sharing with other VMs

Normal hardware access for boot

- Xen virtual firmware runs as VMX root operation with VT-x
- Simulates BIOS expected by OS on initial startup

I/O support

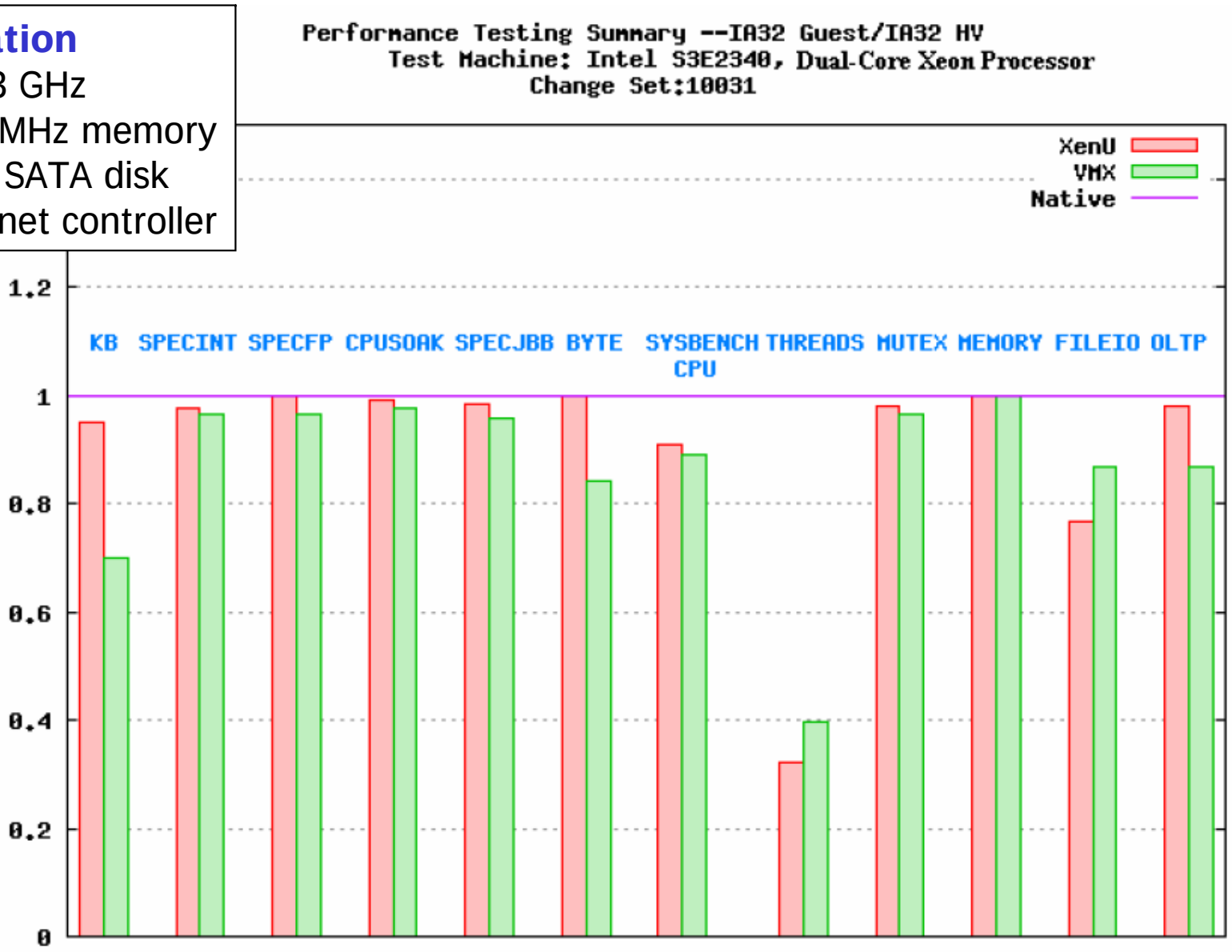
No special drivers

Domain 0 runs Qemu-dm daemon for each HVM Guest

- Supports Domain U HVM Guest for networking and disk access requests

Xen PV and HVM Performance

Test Configuration
Intel Xeon @ 2.3 GHz
4 GB DDR2 533 MHz memory
160 GB Seagate SATA disk
Intel E100 Ethernet controller



Dong, et. al., "Extending Xen with Intel Virtualization Technology", Intel Technical Journal

I/O Bottleneck

Bottleneck — Single Ethernet controller

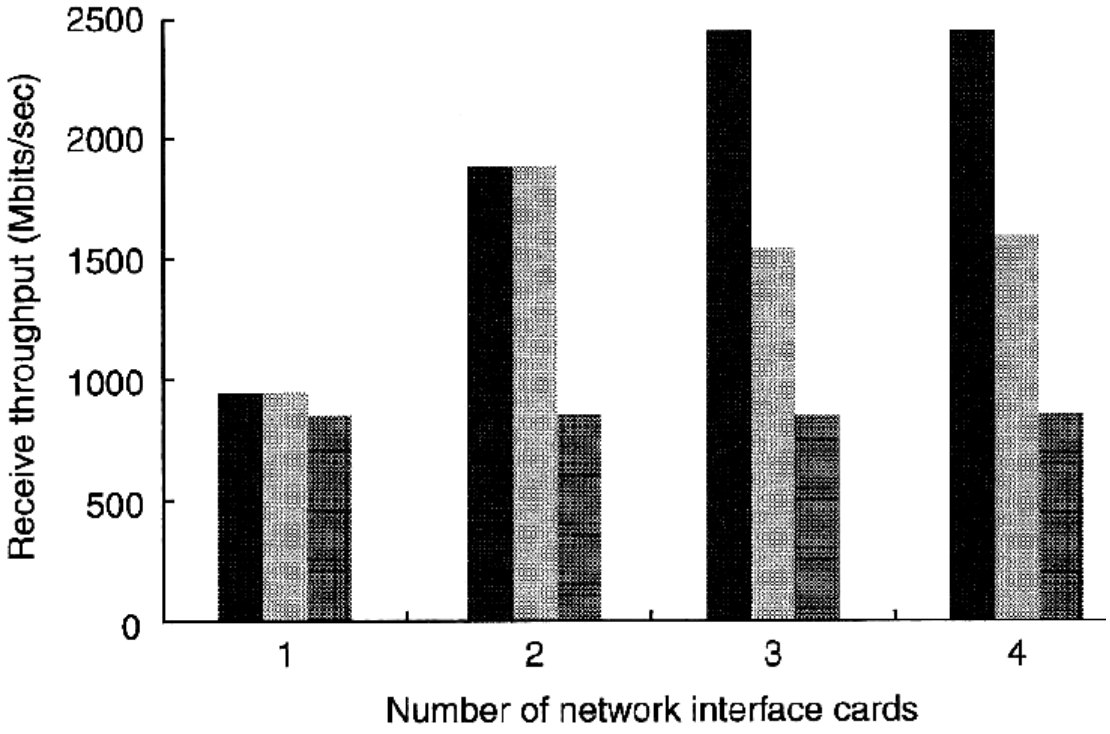
Guest OS tasks waiting for I/O access

hides performance degradation caused by virtualization

Web server running over native Linux without Xen
Threads compete above 2.5 Gbps

Web server running over XenLinux in Domain 0
Threads compete above 1.9 Gbps

Web server running over XenLinux in Domain U PV
Threads compete above 0.9 Gbps



What is Cloud Computing?

Outsourcing service model

- Customer obtains computing services from service provider
  - Service Level Agreement (SLA) guarantees service to customer
- Provider handles operations+administration+maintenance (OAM)
- Business advantages to customer
  - Economies of scale to large provider
  - Cuts labor/capital costs from balance sheet → happy investors

Based on standard technologies

- Cloud service organized from conventional resources
  - Hardware + software + network
  - Provider offers menu of services
- Not a fundamentally different computing technology

Unique technological issues

- Service reliability — provider committed to SLA
- Optimization of provider-side resource configuration
- Optimization of user-side resource configuration

Service Hierarchy in Cloud Computing

Infrastructure as a service (IaaS)

User sees virtual hardware environment

Real hardware or hypervisor / system virtual machine

User installs OS → installs software → runs jobs

Platform as a service (PaaS)

User sees virtual OS environment

OS on single hardware platform or virtual OS

User installs software → runs jobs

Software as a service (SaaS)

User sees virtual application software environment

Applications running on private OS or "sandboxed" on shared OS

Sandbox — private execution environment per application instance

User runs jobs

Storage as a service (STaaS)

User sees virtual mounted storage device

Centralize → Decentralize → Centralize → ?

1950s — 60s

Centralized mainframe computer + multiple OS instances over hypervisor

Timesharing OS serves multiple users

User sees OS environment via dumb terminal (thin client)

1970s

User applications offloaded to minicomputers + timesharing services

User sees timeshared OS environment via dumb terminal

1980s

User applications offloaded to personal workstations (PC)

User sees single-user OS environment running locally

1990s

Network single user workstations

User sees single-user OS environment running locally

2000s

Centralized control of local OS environment by IT departments

2010s

Cloud + netbook / tablet / smart phone — dumb terminal with high-res GUI

Issues in Cloud Computing

Cost

- Provider issues
  - Economies of scale $\Rightarrow$ lower cost per compute job
- User issues
  - Capital + OAM costs $\rightarrow$ operating costs
  - Lower start-up costs $\Rightarrow$ operating debt

Reliability

- Provider issues
  - Redundant infrastructure $\rightarrow$ continuity + disaster recovery
  - Centralized management of OAM, security, performance
  - Virtualization $\rightarrow$ serve multiple users on physical server
  - Multitenancy $\rightarrow$ provide multiple sandboxed application instances on OS
- User sees guaranteed service

Agility

- User / provider reconfigure service / infrastructure as needed
- Growth, load balancing, time-zone serving

Cloud Ownership

Public cloud

- Service provider as public utility — sells / rents computing service
  - Initial providers leverage large existing infrastructure
  - Amazon, Microsoft, Google, IBM
- Menu of services at fixed prices

Private cloud

- Cloud infrastructure for private organization
- Managed internally or outsourced
- Isolates service developers from implementation issues
  - Standard development platform
- Requirements for economic justification
  - Large organization
  - Technology-based services
  - Frequent new service
  - Example — internet content provider