Server Virtualization Inside Out

Jie Lu
CMG’09 workshop

12/04/2009
Overview

- Basis concepts
- Use cases
- Server virtualization types
- Commercial solutions
What is Server Virtualization?

Virtualization
- The pooling and abstraction of resources in a way that masks the physical nature and boundaries of those resources from the resource users
- Enables the sharing of a hardware resource among a number of other software systems

Server Virtualization
- The masking of server resources, including the number and identity of individual physical servers, processors, and operating systems, from server users.
- The implementation of multiple isolated virtual environments in one physical server.
Why Server Virtualization?

- **Reduce total cost of ownership (TCO)**
  - Make better use of resources through higher utilization
  - Save the cost for power, space, cooling, personnel, and etc.

- **Improve business agility**
  - Enable dynamically and automatically re-allocating resources
  - Provide high availability through isolating computing environments
  - Increase flexibility and resiliency through rapid provisioning
Server Virtualization Concepts

- **Hardware abstraction**
  - Hardware Abstraction Layer (HAL) in modern operating systems
    - Hides differences in hardware from most of the operating system kernel
    - All drivers and software talk to the hardware in a unified format
  - Virtual machine (VM)
    - Virtualization makes a physical computer function as if it were multiple virtual computers
    - Each VM is provided with the same basic architecture as that of a generic physical computer, such that it can run or host a guest operating system

- **Guest operating system**
  - An operating system running in a virtual machine environment that would otherwise run directly on a separate physical system
  - A guest OS may or may not be ware of that it is running in a VM
General Architecture

- Stacked layers

![Stacked layers diagram](image)
General Architecture

- Type-1 hypervisor vs. Type-2 hypervisor

Diagram:

- Type-1 hypervisor:
  - Hypervisor
  - Virtual Machine
  - Guest OS
  - App

- Type-2 hypervisor:
  - Host OS
  - Hardware
## Use Cases

### Reasons to virtualize

<table>
<thead>
<tr>
<th>Reason</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut costs via server consolidation</td>
<td>81%</td>
</tr>
<tr>
<td>Improve disaster recovery and backup plans</td>
<td>63%</td>
</tr>
<tr>
<td>Provision computing resources to end users more quickly</td>
<td>55%</td>
</tr>
<tr>
<td>Offer more flexibility to the business</td>
<td>53%</td>
</tr>
<tr>
<td>Provide competitive advantage</td>
<td>13%</td>
</tr>
</tbody>
</table>
Use Cases

- **Server Consolidation**
  - Run multiple production applications while sharing physical hardware resources
  - Reduce server footprint
    - Reduce hardware costs
    - Potentially reduce software costs
    - Reduce power consumption, and cooling demands
    - Free up spaces
  - Easier administration
  - Simplified manageability
Use Cases

Virtual appliance

- A pre-built, pre-configured, ready-to-run application solution packaged along with an optimized operating system (JeOS).
  - Much smaller footprint
  - Requires fewer patches
  - More secure and easier to maintain
- Simplifies software development, distribution, deployment, and management
- Accelerates time to value for customers
- Instantly leverage key capabilities of the underline infrastructure
Use Cases

Hosting legacy applications
- Hardware upgrade and new OS releases are rolled out every 2 – 3 years
- Most OS vendors support only two major releases back
- Old operating systems are not supported by modern hardware or can not take the advantage of the new hardware
- High cost to port legacy applications
- Virtual machine may simulate old hardware
- Takes advantage of increased performance of new hardware
Use Cases

Test and development

- Challenge of the platform supporting matrix
- Multiple environments on a single server
  - Support many operating system releases with various patch levels
  - Extended matrix of OS-middleware combinations
- Accelerated application rollout
  - New applications are validated inside virtual environment
  - Roll out application together with its environment
Use Cases

- Disaster recovery
  - DR by duplication
    - Duplicate entire infrastructure at recovery site thousands miles away
    - The recovery site has to remain identical to the production site
    - Double the cost
  - Each virtual server is a self-contained, hardware independent virtual machine
    - No need to keep identical hardware at recovery site
  - An entire virtual server can be turned into just a few files
    - Recovery is as simple as recovering those files
Use Cases

- **Hardware maintenance and server refresh**
  - High availability for planned interruption
  - Migrate virtual machine before shutting down host
  - Migrate virtual machine back after upgrade
  - Minimize down time

- **Load balancing**
  - Spread work among multiple physical servers
  - Moving virtual server is much easier than migrating application
Use Cases

- **Dynamic provisioning**
  - Server provisioning is about making a system ready for operation
    - Labor intensive and time consuming
  - Defining server configuration based on organizational requirements
  - Pool available servers
  - Select from the pool a server with appropriate software loaded
  - Virtual servers provide flexible infrastructure and rapid provisioning
Server Virtualization Classifications

- **CPU usage**
  - Dedicated vs. shared

- **Virtualization layer location**
  - Type 1 hypervisor vs. type 2 hypervisor

- **Architectures**
  - Hardware partitioning
  - Full virtualization
  - Para-virtualization
  - Emulation
  - OS-level virtualization
Hardware partitioning

- Create multiple isolated partitions on a single server
- Each partition is assigned dedicated processors, memory and I/O channels
- Each partition runs an independent instance of operating system
- Separated management console
- Dynamic partitioning can change the configuration while the system is running

Examples
- IBM Logical Partitions (LPAR / DLPAR)
- Sun Dynamic System Domains (DSD)
- Sun Logical Domains (LDOM)*
- HP nPartitions (nPar) and virtual partitions (vPar)
## Hardware partitioning

<table>
<thead>
<tr>
<th></th>
<th>IBM</th>
<th>HP</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LPAR</strong></td>
<td>Power-based server</td>
<td>Power-based server</td>
<td></td>
</tr>
<tr>
<td><strong>DLPAR</strong></td>
<td>Power-based server</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>nPar</strong></td>
<td>HP 9000 or Integrity Server</td>
<td>HP 9000 or Integrity Server</td>
<td>Enterprise class server</td>
</tr>
<tr>
<td><strong>vPar</strong></td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
</tr>
<tr>
<td><strong>DSD</strong></td>
<td>HMC</td>
<td>HMC</td>
<td>SSP/SC</td>
</tr>
<tr>
<td><strong>LDOM</strong></td>
<td>SC</td>
<td>SC</td>
<td>Control Domain</td>
</tr>
<tr>
<td><strong>Partition type</strong></td>
<td>Static</td>
<td>Dynamic</td>
<td>Dynamic</td>
</tr>
<tr>
<td><strong>Boundary</strong></td>
<td>Processor</td>
<td>Processor</td>
<td>System Board</td>
</tr>
<tr>
<td><strong>Operating system</strong></td>
<td>AIX</td>
<td>AIX, Linux</td>
<td>HP-UX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP-UX, Linux, Windows</td>
<td>Solaris</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Solaris</td>
</tr>
</tbody>
</table>
Full Virtualization

- Also called native virtualization or hardware-level virtualization
- The virtualization layer exports the virtual machine abstraction identical to a physical machine
- Many instances can be run at once
- Virtual machine has the same CPU as the physical server
- Support unmodified guest OS
  - Hypervisor intercepts the privileged operating system calls and service those calls using an emulation of underlying hardware resources.
  - Higher overhead

- Examples
  - VMware ESX Server
  - Citrix XenServer (with Intel-VT or AMD-V)
  - Microsoft Hyper-V (with Intel-VT or AMD-V)
Para-virtualization

- A technique that presents a software interface to virtual machines that is similar but not identical to that of the underlying hardware

- Requires guest OS to be explicitly modified and ported to run
  - Privileged instructions are replaced or trapped on a static basis.
  - Lower overhead

- No modifications are required to guest applications
  - Hypervisor virtualized all architectural features required by the existing standard ABI

- Usually employs virtual I/O server model

- Examples
  - Citrix XenServer
  - IBM Micro-partitions (SPLPAR)
  - Microsoft Hyper-V
## Para-virtualization

<table>
<thead>
<tr>
<th></th>
<th>CITRIX</th>
<th>IBM</th>
<th>Sun Microsystems</th>
<th>Microsoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>XenServer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Micro-partition (SPLPAR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical Domains*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyper-V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Chipset</td>
<td>x86</td>
<td>Power5</td>
<td>UltraSPARC T1</td>
<td>Intel VT, AMD-V</td>
</tr>
<tr>
<td>Hypervisor</td>
<td>Software</td>
<td>Firmware</td>
<td>Firmware</td>
<td>Software</td>
</tr>
<tr>
<td>Management Console</td>
<td>Domain0</td>
<td>HMC</td>
<td>Control Domain</td>
<td>Parent Partition</td>
</tr>
<tr>
<td>Device Driver</td>
<td>Driver Domain</td>
<td>Virtual I/O Server</td>
<td>Service Domain</td>
<td>VSP</td>
</tr>
<tr>
<td>Guest OS</td>
<td>Linux, Solaris</td>
<td>AIX, Linux</td>
<td>Solaris</td>
<td>Windows Server 2008</td>
</tr>
</tbody>
</table>
Emulation

- Also called hosted virtualization or type-2 hypervisor
- Partitioning and virtualization services run on top of a standard operating system (the host)
- The virtualization software relies on the host operating system to provide the services to talk directly to the underlying hardware
- The simulated virtual hardware is independent of the nature of the host computer

Examples
- Microsoft Virtual Server
- VMware Server (formerly GSX Server)
- HP Integrity Virtual Machines (IVM)
Operating System-level Virtualization

- Partition one OS instance into multiple isolated environments
  - Each guest has its own network identities
  - Processes in different guests interact only through network connection
- The virtualization layer sits between the operating system and the application programs
- The guest shares the same OS kernel of the host
  - Low CPU and memory overhead
- Purely an OS feature
  - No dependencies on hardware
  - May be inside other virtual machine, such as hardware partition
- Examples
  - Solaris Containers (Zones + SRM)
  - AIX Workload Partitions (WPAR)
## Server Virtualization Types

<table>
<thead>
<tr>
<th></th>
<th>Hardware partitioning</th>
<th>Full virtualization</th>
<th>Para-virtualization</th>
<th>Emulation</th>
<th>OS-level virtualization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU use</strong></td>
<td>Dedicated</td>
<td>Shared</td>
<td>Shared</td>
<td>Shared</td>
<td>Shared</td>
</tr>
<tr>
<td><strong>Resource scheduling</strong></td>
<td>Guest OS</td>
<td>Hypervisor</td>
<td>Hypervisor</td>
<td>Host OS</td>
<td>Host OS</td>
</tr>
<tr>
<td><strong>OS kernels</strong></td>
<td>Multiple</td>
<td>Multiple</td>
<td>Multiple</td>
<td>Multiple</td>
<td>Single</td>
</tr>
<tr>
<td><strong>Management console</strong></td>
<td>Separate processor</td>
<td>Virtual machine</td>
<td>Virtual machine</td>
<td>Host OS</td>
<td>Host OS</td>
</tr>
<tr>
<td><strong>Virtualization layer</strong></td>
<td>Hardware</td>
<td>Between hardware and guest OS</td>
<td>Between hardware and guest OS</td>
<td>Between host OS and guest OS</td>
<td>Between OS and application</td>
</tr>
<tr>
<td><strong>Isolation</strong></td>
<td></td>
<td></td>
<td></td>
<td>Increasing</td>
<td></td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td></td>
<td></td>
<td></td>
<td>Increasing</td>
<td></td>
</tr>
</tbody>
</table>
Commercial Solutions

- IBM
  - PowerVM (LPAR, DLPAR, SPLPAR, WPAR)
- HP
  - nPar, vPar, Integrigy Virtual Machine
- Sun
  - Dynamic System Domains, Logical Domains, Containers
- VMware
  - ESX Server
- Citrix
  - XenServer
- Microsoft
  - Virtual Server 2005, Hyper-V
IBM PowerVM

- Virtualization on Power-based platforms (System p and System i)
  - Formerly known as Advanced Power Virtualization (APV)
- Logical Partitions (LPAR)
  - Divide a single server into multiple completely independent virtual servers
  - Each LPAR owns dedicated processors and memory
  - Introduced with Power4 and AIX 5.1
- Dynamic Logical Partitions (DLPAR)
  - Processors, memory and I/O channels can be dynamically moved between partitions without restart the involved OS
  - Introduced with Power4 and AIX 5.2
IBM PowerVM

- Shared Processor Partitions (SPLPAR or micro-partitions)
  - Processors are allocated to a pool
  - SPLPAR runs in Shared Processor Pool
  - Capacity is defined as entitlement at granularity of 1/10 of a CPU
  - SPLPAR owns virtual processors, which has no binding to physical processors
  - Firmware based hypervisor dispatches virtual processor to physical processor according to scheduling rules
  - Introduced with Power5 and AIX 5.3
IBM PowerVM

- **Virtual I/O Server (VIOS)**
  - An LPAR that owns hardware I/O adapters
  - An AIX 5.3-based OS with restricted shell
    - Provide command line interface (VIOS CLI) for management.
  - Allow sharing of physical I/O resources between LPARs
  - Provide virtual storage and network adapter capability to client LPARs
    - Virtual SCSI (VSCSI)
    - Shared Ethernet adapter (SEA)
    - Virtual LAN (VLAN)
IBM PowerVM

- Management via Hardware Management Console (HMC) or Integrated Virtualization Manager (IVM)
  - HMC is a dedicated workstation for configuring and operating System p (formerly pSeries) servers
    - Accessed via ssh
  - IVM runs on the first LPAR, which is a VIO Server that owns all the physical adapters
    - Accessed mainly by the IVM website from this VIO Server
  - Create and store partition profiles
  - Start, stop and resetting partition
IBM PowerVM

- **Active Shared Memory**
  - Power6 feature that allows memory over committing
  - Config a shared memory pool, and assign partitions to it
  - Memory can be reallocated from one partition to another

- **Live Partition Mobility**
  - Power6 feature
  - Migrate running AIX and Linux partitions and their hosted applications from one physical server to another without disrupting the infrastructure services
  - Maintains complete system transactional integrity
    - Transfers the entire system environment, including processor state, memory, attached virtual devices, and connected users.
IBM AIX Workload Partitions (WPAR)

- AIX 6.1 feature
- Separate regions of application space within a single AIX image
  - Each WPAR is a separate administrative and security domain
  - Each WPAR obtains a regulated share of system resources

- Shared system resources
  - Operating system
  - Processors
  - I/O devices
  - Shared library and text

- WPAR types
  - System
  - Application

- Application mobility
HP Partitioning

- **nPartitions (nPar)**
  - Provide complete electrical and software isolation
  - Each nPartition owns one or more cells (with processors, memory and I/O channels) exclusively
  - nPartitions can be reconfigured without modifying complex, but the OS has to be restarted*
  - Support different OS
    - 9000 Servers
      - HP-UX 11i
    - Integrity Servers
      - HP-UX 11i, Windows Server 2003, RHEL, SLES
HP Partitioning

Virtual Partitions (vPar)

- Run multiple instances of HP-UX 11i within a single server or nPartition
- Each vPartition owns processors and memory exclusively
- Support dynamic CPU and memory migration
- A nPar can be further partitioned into multiple vPars
HP Integrity Virtual Machines (IVM)

- Create multiple virtual machines on top of an HPUX 11i
  - Each virtual machine is represented by a process
- Virtual machine owns virtual CPU, memory and devices
- Each virtual machine can be configured for up to four virtual CPU
- Increase utilization and scalability by sharing physical resources
- Dynamic resource allocation based on demand and entitlement

- Supported OS
  - HP-UX 11i v2
  - Windows 2003
HP Virtualization

Web Server
Shipping Department
Database Server 1
Database Server 2
Other Users

HP Global Workload Manager

PRM Group A
PRM Group B
Integrity VM Guest OS
Integrity VM Guest OS

memory
memory
memory
memory

core core core core
core core core core
core core core core
core core core core

vPar 1
vPar 2
Integrity VM Host

nPartition 1
nPartition 2

Hardware Platform
Sun Dynamic System Domains (DSD)

- Partition the enterprise-class server to run multiple OS instances
  - Provide complete software isolation based on system boards
  - Each domain owns dedicated resources

- Service Console (SC, formerly System Service Processor)
  - A separate management console

- Dynamic reconfiguration (DR)
  - Change configuration online
  - Solaris 8 and above
Sun Logical Domains (LDom)

- Para-virtualization solution for UltraSPARC T1/T2 based systems (Sun Fire T1000/T2000)
  - Lightweight hypervisor in firmware
  - Require modifying guest OS kernel

- Each virtual machine appears as an independent machine
  - Assigned with dedicated threads
  - Run separate operating systems

- LDom Dynamic Reconfiguration
  - Resize VM on the fly
  - Solaris 10

- Domain roles
  - Control domain
  - I/O domain and Service domain
Solaris Containers

- OS-level virtualization technology available in Solaris 10
  - Single OS image
- **Container = zone + resource manager**
- **Zone**
  - Act as completely isolated servers within a server
    - Each zone has its own identity
  - Each zone has security boundary
    - Processes in different zones interact only through network connection
  - Global zone contains non-global zones
Solaris Containers

- Resource manager
  - Policy-based control of resources
  - Fair-share scheduler (FSS)
  - Projects and tasks

- Resource pool
  - Resource pool is processor set (pset) + scheduling class
  - Default pool and default pset
  - Zone or Project can be assigned to a resource pool
VMware ESX Server

- Virtualization for x86 based servers
- Provide a neutral virtual hardware to Virtual Machine (VM)
  - CPU, Memory, Disk, NIC
- ‘vmkernel’, a specialized Linux kernel
- Use share based resource allocation
- Service Console (COS) provides management interfaces
  - Obsolete
VMware vSphere (Virtual Infrastructure)

- **ESX Server**
- **vCenter Server (Virtual Center)**
  - monitor and manage multiple ESX or GSX servers
- **Vmotion**
  - Transfer virtual machines between servers on the fly
- **Virtual SMP**
  - Up to 4 virtual CPU for guest OS
- **High Availability**
- **Distributed Resource Scheduler**
Citrix XenServer

- **Virtual machine monitor (VMM) for x86**
  - Originally developed by the Computer Laboratory at University of Cambridge

- **Para-virtualization for traditional x86**
  - Requires the modifications on the guest OS kernel (XenLinux)
  - No modification on guest applications
  - Guest OS run at a lower privilege level
  - Privileged instructions are validated and executed within hypervisor

- **Support full virtualization for Intel VT or AMD-V**
  - No modification is required on the guest OS kernel

- **Device drivers are outside of hypervisor, in Driver Domain**
Citrix XenServer

Architecture

VM0
Device Manager & Control s/w
GuestOS (XenLinux)
Native Device Drivers

VM1
Unmodified User Software
GuestOS (XenLinux)
Front-End Device Drivers

VM2
Unmodified User Software
GuestOS (XenLinux)
Front-End Device Drivers

VM3
Unmodified User Software
GuestOS (WinXP)
Front-End Device Drivers

AGP
ACPI
PCI
x86_32
x86_64
IA64

Xen Virtual Machine Monitor

Hardware (SMP, MMU, physical memory, Ethernet, SCSI/IDE)
Microsoft Virtual Server 2005 R2

- A system service running on top of Windows Server 2003
  - Each VM runs in its own thread
  - I/O occurs in child threads
- Host OS kernel schedules CPU execution
- Devices are accessed via host OS device driver
- Guest OS requires no modification
  - Windows NT/2000/XP/2003 (32-bit only)
  - Linux and other x86 based OS can be guest OS, and will be officially supported
Microsoft Hyper-V

- Hypervisor-based virtualization solution
  - Requires Intel VT or AMD-V
- Parent partition
  - A virtual machine
  - Run minimum footprint Windows to manage the child partitions
- Virtualization stack
  - Collection of components that run in the parent partition for VM management
  - Interact with hypervisor
- Child partitions
  - Virtual machines
- Enlightenments
  - Modifications to an OS to make it VM aware
  - Enable optimizations such as memory manager
Microsoft Hyper-V

- No built-in driver model
  - Drivers run in a partition
  - Leverage the large base of Windows drivers
  - Virtualization service providers (VSP)
    - Run within the partition that owns the corresponding physical device
    - Virtualize a specific class of device by exposing an abstract device interface
  - Virtualization service clients (VSC)
    - Consume virtualized hardware service
  - VMBus
    - Enables VSPs and VSCs to communicate efficiently using memory sharing and hypervisor IPC messages
CPU Virtualization
CPU Virtualization

- **Simultaneous Multithreading**
  - Fetch instructions from multiple threads in a cycle
  - Duplicate architectural state registers
  - A physical processor (core) is presented as multiple logical processors to OS
  - Designed for Throughput Computing
  - Difficult to measure utilization
  - Examples
    - Intel Pentium 4 HyperThreading (HTT)
    - IBM Power5 SMT
    - Sun UltraSPARC T1 CoolThreads (CMT)
CPU Virtualization

- CPU virtualization in traditional OS
  - Multiprogramming and multitasking
    - CPU execution switches among many jobs
      - Increase CPU utilization
    - Many users or applications share the computer simultaneously
    - OS schedules CPU
      - Make decision to run a job among those are ready to run
  - Interrupt driven OS
    - Events are signaled by the occurrence of an interrupt or trap
    - OS handles the interrupt
  - Dual mode operation
    - User mode vs. kernel mode
CPU Virtualization

Properties of virtual machine environment

- Equivalence
  - A program running under the virtual machine should exhibit a behavior essentially identical to that demonstrated when running on an equivalent physical machine directly.

- Resource control
  - The hypervisor must be in complete control of the virtualized resources.

- Efficiency
  - A statistically dominant fraction of machine instructions must be executed without hypervisor intervention
CPU Virtualization

- Popek and Goldberg virtualization requirements
  - Classification of instructions
    - Privileged instructions
      - Those that trap if the processor is in user mode and execute if it is in kernel mode.
    - Control sensitive instructions
      - Those that attempt to change the configuration of resources in the system.
      - Updating virtual to physical memory mapping
      - Communicating with devices
      - Manipulating global configuration registers
    - Behavior sensitive instructions
      - Those whose behavior or result depends on the configuration of resources (the content of the relocation register or the processor's mode).
      - Load and store operations that act on virtual memory
CPU Virtualization

- Popek and Goldberg virtualization requirements
  - A virtualizable architecture requires that the set of sensitive instructions for that computer is a subset of the set of privileged instructions.
    - All instructions that could affect the correct functioning of the VMM always trap and pass control to the VMM
  - Non privileged instructions must be executed natively for efficiency

- x86 problems
  - 17 instructions violate the virtualization requirements
  - Protection system instructions
    - LAR and LSL
  - Sensitive register instructions
    - SIDT
CPU Virtualization

Virtual machine

User process

Virtual user mode

Guest OS

Virtual kernel mode

Hardware

user mode

kernel mode
Binary Rewriting

- Emulate privileged instructions
- Hypervisor scans the instruction stream and identifies privileged instruction
  - These are rewritten to point to their emulated versions
  - Jump instruction starts another instruction stream scan
- Similar to debugger implementation
  - Insert breakpoints at any unsafe instruction and any jump
  - Utilize hardware support for debugger
Hypervisor Call

- System call in traditional OS
Hypervisor Call

- Decrease guest OS privilege level
  - Guest has to deal with the consequences from instructions that doesn’t trap
  - Replacement of privileged instructions occur at compile time

- Interrupt driven hypervisor call
  - Guest kernel generates interrupts for hypervisor to handle
    - Xen use interrupt 82h for hypervisor calls

- Indirect hypervisor call via memory address
  - A hypervisor call page contains the code, and is mapped into the guest’s address space
  - Hypervisor calls are issued by calling an address within the page
Hypervisor Call

Native

Ring 3
Application

Ring 2
System call

Ring 1
Kernel

Ring 0
Kernel

Virtualized

Application

System call

Fast system call

Kernel

Hypervisor

Hypervisor call
Hardware Assisted Virtualization

- Intel VT and AMD-V
- Additional higher privilege mode (ring -1)
  - Guest OS remains in ring 0, while hypervisor resides in ring -1
- Tagged translation lookaside buffer (TTLB)
  - 6-bit Address Space ID (ASID) associated with TLB
- Shadow Page Tables
  - Trap page table modification to hypervisor
- Nested Page Tables
  - Add another layer of indirection to virtual memory
  - Four-level page table
    - First layer identified by a Virtual-Processor Identifier (VPID)
Booting Up Virtual Machine

- **Bootstrapping in standalone computers**
  - Bootstrap loader
    - Resides in firmware
    - Runs diagnostics and initializes all aspects of the system
    - Reads bootstrap program from boot block into memory and executes the code
  - Bootstrap program
    - Resides in boot block, which is a single block at a fixed location on disk
    - Locates the kernel, loads it into memory, and starts its execution

- **Bootstrapping hypervisor**
  - Same as booting regular OS
    - Install interrupt handlers
  - Starts the management console VM immediately afterwards
Booting Up Virtual Machine

- Emulated BIOS
  - Store BIOS configuration in a file
    - VMware uses nvram file
  - VMM loads configuration file and starts its bootstrap loader
Booting Up Virtual Machine

- Booting up as paravirtualized guest
  - BIOS is replaced by a variety of different facilities
    - Xen uses
      - Start info page, contains basic information required by a guest to initialize the kernel
      - Shared info page, gives some more data and is updated while the guest is run
      - XenSotre, determines available virtual devices
  - Install interrupt handler
    - Most interrupts will be passed on to any guest that requests them
    - Some write a jump address into the interrupt vector
    - Some bind an event port to a virtual or physical IRQ
Time Keeping in Virtual Machine

- Time keeping accuracy is important
  - Wall-clock time, the real time that has elapsed
    - For user space applications that run scheduled tasks, display clocks, and etc.
    - For time stamping events (file system, log)
  - Virtual time, the amount of time the system has spent running
    - Essential for resource scheduling and accounting
  - Wall-clock time and virtual time matches on standalone system

- Timer
  - Implemented by a fixed-rate clock and a counter
  - Counter decrements for every clock tick
  - Interrupt is generated when counter reaches 0
  - Instructions that modify the content of the timer are privileged
Time Keeping in Virtual Machine

- OS keeps track of time by counting timer interrupts or ticks
  - At boot time, reads initial time from CMOS or queries network time server
  - Then, sets up timer to interrupt periodically at a known rate
  - Afterwards, updates time fields based on timer interrupts

- Virtual time does not match wall-clock time in virtual machine
  - Guest OS is not running all the time

- Time keeping accurately in a virtual machine is difficult
  - A VM may not be running at the moment it should generate a time interrupt
  - A VM may not check for the interrupt at the moment it is generated
Time Keeping in Virtual Machine

- **VMware approach**
  - Keep track of the current timer interrupt backlog
  - Deliver timer interrupts at a higher rate whenever the backlog gets too large
    - Lost tick
  - Give up catching up if backlog of interrupts grows beyond 60 seconds
    - Set backlog to zero
    - VMware Tools time synchronization feature kicks in
      - Synchronize the guest OS time to match the host machine's clock
    - Resume keeping track of backlog
  - **Scalability issue**
    - Excessive context switches for timer interrupts
Time Keeping in Virtual Machine

- **Xen solution**
  - Shared info page
    - Mapped into guest’s address space at boot time
    - Updated dynamically
    - Used throughout the runtime of a guest kernel to retrieve global state
    - Keeps Wall-clock time and virtual CPU time information
  - Keep track of three time values
    - Initial system time, the time of day when system time is zero
    - Current system time, the time that has elapsed since the virtual machine is resumed, and is updated whenever the guest is scheduled
    - Time-Stamp Counter (TSC) time, the number of cycles that have elapsed since and arbitrary point in the past
      - Get from TSC register with RDTSC instruction
Time Keeping in Virtual Machine

- Translate a TSC difference into nanoseconds
  • \((tsc \ll timeinfo->tsc_shift) \times timeinfo->tsc_to_system_mul\)

- Calculate current system time
  • TSC time determines how much time has elapsed since the system time stamp was written
  • Every time system time is modified by the hypervisor, the TSC value is also written \((vcpu_time_info_t->tsc_timestamp)\).

- Compute current wall-clock time
  • Add current system time to initial system time (kept in \(wc\_sec\) and \(wc\_nsec\) in shared info page)
CPU Scheduling

- CPU scheduling in virtualized environment is similar to multithreading scheduling in regular OS
  - Guest OS schedules processes/threads to virtual processors
    - Threading library maps user space threads to kernel threads
    - Guest OS schedules kernel threads to virtual processors
  - Hypervisor schedules virtual processors to physical processors
CPU Scheduling

- Most hypervisors use Fair Share Scheduler
  - Each VM is assigned CPU share
  - Each VM can be viewed as a fair share group
  - Each virtual processor can be viewed as a process

- Many hypervisors use combination of reserved capacity and FSS
  - Each VM has a reserved capacity, i.e. entitlement
    - Reserved capacity is guaranteed to the VM, whenever it needs
    - The sum of all reserved capacity cannot exceed host capacity
  - Beyond reserved capacity, VM’s compete under FSS
Memory Management
Memory Management in Operating Systems

- Contiguous memory allocation
  - Each process is contained in a single contiguous section
  - MMU maps logical address dynamically
  - OS maintains information about allocated partitions and free partitions (hole)

- Paging
  - Permits the physical address space of a process to be noncontiguous
    - Virtual address space vs. physical address space
  - Page table contains the base address of each page in physical memory

- Segmentation
Memory Management in Virtualization

Application

Kernel

Hypervisor

Virtual address

Logical address

Physical address
Memory Partitioning

- Guest VMM undergoes major changes
- Logical Memory
  - An abstract representation that provides a contiguous memory address to a VM or partition
  - Provides the isolation and security of the guest OS from direct access to physical memory
  - Guest OS handles the logical memory as if it were physical memory in a non-virtualized environment
Memory Partitioning

- Physical-to-logical memory mapping
  - The whole physical memory is divided into physical memory blocks (PMB)
    - PMBs assigned to a VM need not be contiguous
  - The logical memory is divided into logical memory blocks (LMB)
  - Reserved memory for hypervisor
Memory Partitioning

- Hypervisor has access to the entire memory space
- Global partition page table
  - Consists of the mapping of PMBs to the LMBs of different partitions
  - Maintained by hypervisor
    - Guest OS can access it only through hypervisor calls
    - Ensures that partitions do not access the memory of another partition

- Address translation
  - Guest OS VMM requests the hypervisor to convert a virtual address to a correct logical address
  - Hypervisor converts a virtual address to a system-wide physical address using the global partition page tables
  - Hypervisor translates the obtained system-wide physical address (PMB) to the corresponding valid logical address (LMB) for the partition
Memory Reclamation

- Over-allocate memory in OS
  - Increases degree of multiprogramming
  - Demand paging saves space for pages that are never referenced
  - Page replacement determines which frames to be paged out to swap space

- Memory over-commitment in virtual server
  - No memory over-commitment is allowed
  - Hypervisor-level paging
    - Transparent to guest OS
    - Hypervisor makes page replacement decision
      - May collide with guest OS paging policy
    - Possible double paging
Balloonning

- **Force guest OS paging first**
  - Guest OS has better knowledge to make best page replacement decision

- **Balloon module**
  - Loaded into the guest OS as a pseudo-device driver or kernel service
  - No external interface within the guest
  - Communicates with hypervisor via private channel

- **Inflate (increase pressure)**
  - Driver allocates pinned physical pages with the VM (logical page frames)
  - Hypervisor reclaim the backing physical page frames
  - Guest OS paging may kick in under memory pressure

- **Deflate (decrease pressure)**
  - Driver releases previously-allocated pages to make them available to VM
Ballooning

- Reclaim memory (inflate balloon)
- Allocate more memory (deflate balloon)
- Guest OS may page out to virtual disk
- Guest OS may page in from virtual disk
Memory Sharing in OS

- **IPC shared memory**
  - Producer-consumer model

- **Memory-mapped file**
  - Uses virtual memory techniques to treat file I/O as routine memory accesses

- **Copy-on-Write**
  - For fast process creation
Memory Sharing in Virtual Server

- Motivation
  - Multi VM run same OS and applications
  - Increase scalability

- Explicit shared memory in hypervisor
  - For inter-VM communication
  - Similar to IPC shared memory
  - Producer-consumer model
  - XenServer uses grant table

- Implicit shared memory
  - Zeroed pages
  - Copy-on-Write
  - Content-based scanning (used by VMware ESX Server)
Memory Sharing in Virtual Server

- Content-based scanning

VM 1  VM 2  VM 3

Physical Memory

Hash: ...12af
Refs: 2
MPN: 678d

shared frame

hash table

011010
110101
010111
101100

hash page contents

...2d8b12af
Virtual Address Space Layout

- Memory split in OS
  - Map the entire kernel address space into all processes’ address space
    - Eliminates the context switches when moving between user and kernel space
    - Linux uses 1GB/3GB split
    - Windows uses 2GB/2GB split

- Memory split in paravirtualized guest OS
  - Xen reserves top 64MB for hypervisor
I/O Virtualization
Virtualizing I/O Devices

- The number of VMs could be supported in a virtualized computer is limited by the number of physical adapters, or PCI slots available
  - Each VM needs its dedicated I/O channel

- Virtualizing I/O devices involves sharing of I/O adapters among VMs
  - Needs mechanism to multiplex the I/O devices
  - VMs are provided with abstraction of I/O devices
    - Guest OS I/O subsystem work with virtual I/O devices
Monolithic Model

- Hypervisor provides emulated forms of simple devices
  - The emulated devices are typically chosen to be common hardware
    - Likely that drivers exist already for any given guest
  - The hypervisor owns all I/O devices
    - Provides the device drivers for the actual physical devices
    - Each guest only accesses the emulated virtual devices
    - Hypervisor multiplexes I/O requests from all guests

- Employed by full virtualization and emulation
  - VMware ESX Server, VMware Server, Microsoft Virtual Server

- Hypervisor is required to be updated for supporting new hardware
  - Updating hypervisor has higher impact and risk than updating OS
    - Affect all VMs
Monolithic Model

- **Performance**
  - Heavy demands on CPU to execute device driver
    - Double device drivers
    - Device driver is in kernel mode
      - Emulating the privileged instructions for guest OS device driver
    - Context switches stress the CPU and its hardware caches
      - Programmed I/O can be more efficient than interrupt-driven I/O
      - Network traffic is especially stressful

- **Supporting the range of hardware available for a commodity server at hypervisor level would not be economy efficient**
  - Most of them are already supported by the common OS

- **Monolithic model is usually not good for I/O intensive systems**
Virtual I/O Server

- **Client-server model**
  - Most paravirtualization solutions employ it to virtualize the I/O devices
    - Guest OS kernel has to be modified in order to run anyway

- **Virtual I/O server**
  - The hypervisor delegates I/O devices to a special VM, the virtual I/O server
  - The virtual I/O server owns and manages I/O devices
    - Actual device drivers reside in the virtual I/O server
  - Other VM send I/O request to virtual I/O server via inter-VM communication mechanism
  - Leverages the large base of device drivers in existing OS
  - Can change underlying device or driver while client VMs continue to execute
  - Usually the control VM is acting as the virtual I/O server by default
The virtual I/O server owns the physical I/O devices and their corresponding drivers.

The backend service handles multiplexing and exposes an abstract device interface.

The client accesses these abstract virtual devices using the virtual device driver, just like they are connected locally.

The virtual bus is a well-defined protocol built on top of inter-VM communication mechanisms.

The front end interface communicates with the backend service via the virtual bus protocols.
Virtual I/O Server Architecture

Virtual bus

- A well-defined protocol built on top of inter-VM communication mechanisms provided by hypervisor
  - Memory sharing
  - Message passing
- provides a way of enumerating the virtual devices available to a given virtual machine, and connecting to them
Virtual I/O Server Architecture

- **Backend service**
  - A service provider
  - Handles multiplexing
    - Allow more than one client virtual machine to use the device
  - Exposes a generic abstract interface
    - Exports a pool of heterogeneous physical storage as a homogeneous pool of block storage in simple abstract forms
      - The virtualized storage devices can be backed by internal physical disks, external LUNs, optical storage, logical volumes, or even files
  - Implements the Ethernet transport mechanism
  - Implements an Ethernet switch that supports VLAN capability
Virtual I/O Server Architecture

Virtual I/O Server

Client

Application

Virtual device driver

Front end

Hypervisor

Device Driver

Backend
Commercial Examples

- Virtual I/O server terminologies comparison

<table>
<thead>
<tr>
<th></th>
<th>IBM PowerVM</th>
<th>Sun Logical Domains</th>
<th>Microsoft Hyper-V</th>
<th>Citrix XenServer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual I/O server</td>
<td>Virtual I/O Server</td>
<td>Service Domain</td>
<td>VSP Partition</td>
<td>Driver Domain</td>
</tr>
<tr>
<td>Client virtual machine</td>
<td>Client partition</td>
<td>Logical Domain</td>
<td>Child Partition</td>
<td>Domain U</td>
</tr>
<tr>
<td>Virtual bus</td>
<td>Virtual SCSI</td>
<td>Logical Domain</td>
<td>VMBus</td>
<td>XenBus</td>
</tr>
<tr>
<td>Front end interface</td>
<td>Virtual Adapter</td>
<td>Virtual Device Driver</td>
<td>VSC</td>
<td>Top half of split driver</td>
</tr>
<tr>
<td>Backend service</td>
<td>Virtual Server Adapter</td>
<td>Virtual Device Service</td>
<td>VSP</td>
<td>Bottom half of split driver</td>
</tr>
</tbody>
</table>
Commercial Examples

- IBM PowerVM – Virtual I/O Server (VIOS)
  - Part of the System p PowerVM hardware feature
    - Allows virtualization of physical storage and network resources
  - Runs in either a dedicated processor partition or a micro-partition
    - An AIX 5.3-based OS with restricted shell
  - The VIOS is a standard storage subsystem
    - Exports the physical storages in the form of standard SCSI-compliant LUNs
      - VIOS implements the virtual SCSI server adapter to act as a SCSI target device
      - The client logical partitions have a virtual SCSI client adapter as SCSI initiator
    - A client partition can be assigned a whole disk or logical volumes
  - Shared Ethernet Adapter (SEA)
    - Supports link aggregation, failover, TCP segmentation offload, and VLAN.
Commercial Examples

- Sun Logical Domains – Service Domain
  - A single logical domain may function in one or more of the four roles
    - Control domain
    - Guest domain
    - I/O domain
    - Service domain
  - A logical domain channel (LDC)
    - A point-to-point, full-duplex link created by the hypervisor
      - A unique LDC is explicitly created for each link, ensuring data transfer isolation
    - Provide a data path between virtual devices and guest domains
    - Establish virtual networks between logical domains
    - Data is transferred as a simple 64-byte datagram or by using shared memory
Commercial Examples

Citrix XenServer – Driver Domain

- Xen employs a split device driver model
  - Shared ring buffer
    - In shared memory segments
    - The top half driver inserts requests and the bottom half driver places responses
  - The top half of the split driver
    - Initializes a memory page with a ring data structure and exports it via the grant table mechanism
    - Advertises the grant reference via the XenStore
  - The bottom half driver
    - Provides multiplexing features
    - Retrieves the grant reference and maps it into its own address space
- A network driver is added to the Driver Domain kernel as a virtual interface
  - Use existing low-level services for bridging, routing, and virtual interfaces in the guest OS for multiplexing
Commercial Examples

- Microsoft Hyper-V – Virtual Service Provider (VSP)
  - Virtualization service providers (VSPs)
    - Virtualize a specific class of device (e.g. networking, storage, etc.)
    - Expose an abstract device interface
    - Run within the partition that owns the corresponding physical device
    - The physical devices are managed by traditional driver stacks in the VSP
  - Virtualization service clients (VSCs)
    - Consume virtualized hardware service
  - VMBus
    - Software “bus” (enumeration, hot plug, etc.)
    - Enables VSPs and VSCs to communicate efficiently
    - Uses memory sharing and hypervisor IPC messages
Clustering and Mobility
Clustered computers share storage and are closely linked via a LAN or a faster interconnect
- Each node of the cluster usually contains its own operating system
- Clusters offer high scalability and availability with massive parallel processing power

High performance cluster (HPC)
- Involves developing parallel programming applications for a cluster to solve complex scientific problems.

Load balancing cluster
- Distributes the workload evenly over multiple back end nodes
Cluster

- **High availability cluster**
  - Services continue even if one or more systems in the cluster fail
  - Asymmetric
    - One node is in hot-standby mode, while the other is running the applications
    - The hot-standby node does nothing but monitor the active server
    - If the active server fails, the hot-standby node becomes the active server
  - Symmetric
    - All nodes are running applications and monitoring each other
Cross-system Flexibility

- Single-system virtualization greatly improves the flexibility
- Business requirements often demand a more comprehensive view of the entire infrastructure
  - Applications are usually distributed across multiple systems to ensure:
    - Isolation
    - Optimization of global system resources
    - Adaptability of the infrastructure to new workloads
Cross-system Flexibility

- **Why migration**
  - Resource balancing
    - A system does not have enough resources for the workload while another system does
  - New system deployment
    - A workload running on an existing system must be migrated to a new, more powerful one
  - Availability requirements
    - When a system requires maintenance, its hosted applications must not be stopped and can be migrated to another system
Cross-system Flexibility

- **Transfer of a workload between systems is time consuming**
  - Require careful planning and highly skilled people
    - Other workloads on target system might incompatible with it
  - Often cause a significant downtime
    - An SLA may be so strict that planned outages are not tolerated

- **Migrating a VM is much simpler and can be easily automated**
  - VMs are well encapsulated and abstracted
  - Cold migration
    - Moving a powered off VM
  - Hot migration
    - Moving a VM while service is provided, without disrupting user activities
Cold Migration

- Cold migration is also called inactive migration
- Moves the definition of a powered off VM from one system to another, along with its network and disk configuration
  - A VM is essentially definition stored on disk

- The cold migration procedure
  - A new VM is created on the destination system with the same configuration present on the source system
  - Preserve network access and disk data and made available to the new VM
  - Shut down the VM on the source system
  - Remove the VM configuration and free all involved resources on the source system
Cold Migration

- Cold migration in a cluster
  - In a cluster, VMs are usually reside on shared storage, which are visible to all hosts within the cluster
  - Migration simply reassigns ownership of the VM to the destination system
    - Shut down the VM on the source system
    - Start the VM on the destination system
Cold Migration

Cold migration use cases

- Moving a VM from test to production
  - No shared storage in common
- Moving a VM from local to shared disk
  - Added SAN connectivity
- Source and destination hosts are not compatible for hot migration
- Destination host is not in the same data center
  - Clone VM or template to a new data center
  - Off-site backups
Hot Migration

- Also call active migration, or live migration
- A running VM is moved from a source system to a destination system with no disruption of VM operation or user service
- Applications continue to handle their normal workloads during the migration
- The transfer is imperceptible to users
  - The physical memory content of the VM is copied from system to system
  - Disk data transactions, running network connections, user contexts, and the complete environment are migrated without any loss
- Dynamic load balancing
- Zero downtime maintenance
Hot Migration

- **Commercial examples**
  - VMware VMotion
    - A Component of Virtual Infrastructure
    - Introduced in 2003
  - IBM PowerVM Live Partition Mobility
    - Feature of Power6-based System p Servers
    - Introduced in 2007
  - Citrix XenMotion
    - Feature of XenServer 4.0
    - Introduced in 2007
Hot Migration

- Both cold and hot migration transfer the VM’s definition and resource configuration
- Hot migration involves the transfer of active run-time state
  - VM’s memory
  - Hypervisor-level page table
  - Virtual processor state
  - Virtual adapter state
  - Non-volatile RAM (NVRAM)
  - Time of day
  - state of each resource
Hot Migration Phases

- Preparation
  - Cluster management console initiates the migration
    - Virtual center, HMC
  - Identifies target (destination) host
  - Identifies service VMs on the source and destination system (optional)
    - Virtual I/O Server
  - Reserve dedicate migration network segment
    - For copying memory pages
Hot Migration Phases

- Validation
  - Host compatibility check
    - Compatible CPU
      - Same manufacture, same level of SSE instructions, same width (32 bit or 64 bit)
      - CPU frequency, cache size, number of cores/threads don’t matter
    - Compatible hypervisor
  - Configuration checks
    - Use of any host exclusive resource
  - Resource availability checks
    - Whether or not sufficient resources are available on the destination server
Hot Migration Phases

- **Transferring**
  - Destination host is informed it is about to receive a VM
    - Builds the VM’s virtual hardware
  - Pre-copy memory
    - VM’s memory pages are copied to the destination host
      - A bitmap keeps track of page transfers
        - Bit corresponding to transferred page is set
      - Less frequently used data pages transfer first
        - OS, program code segments
    - If VM updates the page after transfer
      - The bitmap bit is reset
      - Invalidates the page copy
Hot Migration Phases

- Monitor progress
  - Memory pre-copy will copy most but not all of the VM’s RAM
    - Running VM changes pages, invalidating them on the destination host
  - Progress is tracked
    - When marginal progress drops to near zero, it’s time to move on

- De-schedule VM
  - VM is taken off the hypervisor run queue
    - No different than if the VM loses its turn to run

- Memory bit map is transferred to the destination host
  - The destination host then knows which pages are valid
Hot Migration Phases

- VM context is sent to destination
  - CPU register values, MMU values, etc.
  - Needed to restore the running state of the VM
- Pending I/O events are sent to the destination host
  - Ensures no loss of keystrokes, screen updates, disk I/O activity, etc.
Hot Migration Phases

- **Switching over**
  - The destination host now has
    - CPU context
    - All incomplete I/O events
    - VM virtual memory images
      - Not entirely up to date
      - Bitmap indicates invalid pages
  - The source host releases it’s exclusive lock on the VM’s files
    - The destination host asserts new locks on the VM’s files
  - Physical switch is notified of VM MAC relocation to a new port via rarp
Hot Migration Phases

- VM is added to the destination hypervisor run queue
  - Scheduled to run like any other VM
- Resumes VM execution before all its memory pages have been migrated
  - Needed pages are background copied from the source host on demand
  - Effectively demand-paging over network
  - Significantly reduces the length of the pause
Hot Migration Phases

- **Housekeeping**
  - Migration completes when the entire memory image of the VM is up to date on the destination host
  - Source host removes all VM traces
    - Reclaim VM’s memory
    - Destroys VM’s virtual hardware
    - Removes VM from host VM roster
  - Destination host cleans up
    - Removes memory bitmap
Mobility-based Clusters

- **High-availability cluster**
  - Zero downtime for scheduled maintenance
    - Use hot migration
  - Minimize downtime caused by host failure
    - Use cold migration
  - Peers in an HA cluster:
    - Hold copies of the entire cluster configuration
      - Each host knows the VM inventory of all other HA cluster peers
    - Monitor the health of other hosts
      - Exchange heart beats
Mobility-based Clusters

- **Load balancing cluster**
  - Dynamically balancing VMs across the cluster
  - Cluster service monitors host loads, VM resource usage and requirements
    - Looks for VMs that are not getting the CPU cycles or memory needed due to the host resource over commit
    - Checks to see if another host has free resources
    - Hot migrates VMs to a more suitable host
VMware Distributed Resource Scheduler (DRS)

- Running on Virtual Center
- VMs are assigned to cluster or resource pool
  - Not assigned to any host statically
- **DRS performs two primary functions**
  - Initial placement
    - Move VM to least busy ESX host before power
  - Dynamic balancing
    - Monitor ESX resource consumption
    - Monitor VM resource needs
    - Recommend/VMotion VMs whose resource needs are not being met
VMware Distributed Resource Scheduler (DRS)

- **DRS automation level**
  - Manual
  - Partially automated
  - Fully automated

- **Affinity**
  - Keeps VMs on the same ESX host
  - Take advantage of shared memory
  - Take advantage of virtual networking performance

- **Anti-affinity**
  - Keeps VMs on different ESX hosts
  - Keeps the same workloads (e.g. Web Server) off the same ESX host
  - Keeps the VM nodes of a HA cluster off the same ESX host
Resource Pool

- Used to delegate resources in a predictable manner
- Its setting can limit the host resources consumed by a pool
  - Impose caps on CPU and memory requests
  - VMs assigned to a pool are restricted by the pool configuration
- Can be at different levels
  - Outside hosts
    - Resources of the pool is not limited by any single host
  - Within host
    - Resources of the pool is bounded by a single host
References

- [http://www-03.ibm.com/systems/power/software/virtualization](http://www-03.ibm.com/systems/power/software/virtualization)
- [http://www.hp.com/go/virtualization](http://www.hp.com/go/virtualization)
- [http://www.microsoft.com/virtualization](http://www.microsoft.com/virtualization)