Presentation 10

Virtual Machines

וירטואליזציה מתייחסת לשימוש בטכנולוגיות מסוימות לבנות מערכת שהתנהגותה החיצונית מחקה מערכת אחרת. מאז 1964 מחשבי mainframe של חברת IBM מציגים סביבת עבודה וירטואלית למשתמשים — המשתמש רואה סביבת עבודה ליחיד (כמו ב- PC) שנוצרת בפעולות מתואמות במערכת גדולה רבת משתמשים. היום וירטואליזציה מבוססת תוכנה גם מהווה תחליף פשוט לפתרונות מבוססים בחומרה והיא מאפשרת השירותים של מערכת רצויה בלי התקנת המערכת האמיתית בהשקעת זמן ומשאבים רבים.

Slide 2 – 4 Virtual Machine (VM)

In a layered model of computation, software and hardware divided into logical layers, an abstraction which conforms to actual functioning to one degree or another. Layer n receives services from server layer n – 1 and provides services to client layer n + 1. The layers interact through a well-defined programming interface.

A **virtual layer** is the emulation in software of some hardware or software layer n. The name derives from the fact that it provides the virtues (characteristics) of layer n in a way that is transparent to layer n + 1. It provides service to layer n + 1 as expected from a real layer n. A virtual layer n can run at some layer m $\neq$ n in the real system.

Two common examples of virtual systems are shown on slide 3. When a **web browser** exchanges data with a web server, the applications perform virtual communication with one another, as if the browser and server were directly connected. The real system that implements this virtual communication consists of the OS in each system, the computing hardware, and the network infrastructure.

Similarly, in **cloud computing**, a customer enters a service level agreement (SLA) with a provider of computing services. The SLA specifies the virtual computing system that the customer's users will see — hardware, software configuration, and performance. The real system assembled by the provider to implement the virtual configuration is invisible to the customer and the user, as long as it meets the requirements of the SLA.

A basic computing system consists of hardware, an OS that manages the hardware and provides user access to hardware resources, and application programs running above the OS. Slide 4 shows three types of virtualization.

In a **Process Virtual Machine**, the VM provides application interpretation above OS — it allows applications designed to interact with the VM to obtain standard services from the OS. These applications can then run above any OS for which an appropriate VM has been written.

In a **Hosted Virtual Machine**, a virtual machine monitor (VMM) runs between the primary OS and a guest OS. The VMM provides the guest OS with software emulation of a real hardware system, so that applications written for the guest OS may run as usual.

In a **System Virtual Machine**, a VMM runs directly above the physical hardware to emulate a system-level hardware environment. The VMM permits one or more OSs to run above the virtual hardware.

Slide 5 Process VM Example — Java

A well-known example of a Process Virtual Machine is Java. The Java VM runs above a standard OS, providing an implementation dependent interface to system hardware. I/O operations are performed by calls to the OS. High level Java programs are compiled to bytecode which usually runs (interpreted) in the Java VM to machine code for the real CPU. In IBM mainframes and other certain other processors Java bytecode can be run as the native machine language, permitting execution without interpretation in the VM.

Slide 6 Hosted VM Example — Guest OS Over OS

In 32-bit versions of Windows, the program cmd.exe provides a DOS command line interface in a window. In order to create this virtual DOS environment, Windows allocates 1 MB of virtual memory space and copies the DOS kernel into low memory. Applications running in this 16-bit DOS window issue system calls that are handled by the guest DOS kernel, but the DOS accesses to hardware are trapped and served by the Windows host OS, which returns system responses to DOS. Concurrent DOS windows can be opened by multiple allocations of a 1 MB virtual memory space. The DEBUG application running on a native (real) DOS system has full access to the all system resources, including hardware registers. In a virtual DOS machine, DEBUG sees the 1 MB memory space allocated by Windows, which emulates register values for DOS.

In a similar way, host virtualization programs such as Parallels (Apple), VirtualBox (Oracle), VMware (EMC), DOSBox (GNU) permit an OS to run in a virtual environment above a host OS.

Slide 7 – 8 Virtual Machine in IBM z/990 Mainframe

The hardware layer of a mainframe computer consists of multiple CPUs, the I/O system, and an internal communication network. A VMM called a hypervisor runs above the hardware, providing an operator console for partitioning/configuring CPUs and I/O, and hardware emulation as an abstraction to the OS layer. The OS for a mainframe expects to interact with the hypervisor (not real hardware). The hypervisor configures the system into logical partitions (LPAR), each of which runs a separate instance of an OS. With multiple LPARs, the system can run z/OS, MVS, VM, Unix, Linux, Windows, and other OS instances in parallel. The user sees a single-user interface provided by one OS.

The mainframe VM serves the administrator as a system management tool that isolates user environments on single hardware platform. It permits multiple OSs or multiple copies of a single OS to run independently, while maintaining higher security. The VM also centralizes and simplifies resource management for hardware redundancy, high availability, recovery management, hardware pooling (dynamic assembly of clusters for efficient mapping of applications to hardware and load balancing).

Slide 9 – 10 Virtualization for Server Systems

As PC-type workstations evolved into application servers around 2000, it was typical to run one application per physical server that was specified to handle the worst-case load. As server farms grew, this led to a large number of underutilized servers. Competition from modern mainframes (with powerful RISC CPUs and a Posix-compatible OS) grew, presenting the advantages of dynamic load balancing, centralized power, cooling, monitoring, and backup, and very high SAR

(scalability, availability, reliability). In general, mainframes can offer lower cost per served client than server farm.

Manufacturers of traditional x86 server systems responded by offering virtualization systems to partition hardware resources for running independent applications. In particular, Intel introduces its Virtualization Technology (VT) to support its IA-32 and IA-64 ISAs, and its I/O chipset. Slide 10 shows the virtual partition of an HP Unix-oriented server system. On the left side, the unpartitioned system includes 4 CPUs and a system bus that connects main memory and a pair of independent I/O systems. This system boots UX (HP's Unix version). On the right, the system boots a virtual partition monitor (VMM) that performs a virtual partition of the hardware, and then boots independent UX OSs on each partition.

Slide 11 – 15 VM Interventions

In a mainframe or other system VM, the hypervisor (VMM) is the lowest layer above physical hardware (the host). It creates virtual machine (VM) environments for OSs and allocates physical host resources to virtual resources. Processor intensive applications run with low VMM overhead — most instructions run directly on hardware and make infrequent OS calls. But I/O intensive applications run with high overhead, making frequent OS calls for emulated I/O services. Some applications are I/O-limited — the program throughput inherently limited by the usual causes of I/O latency and so emulation adds relatively little overhead.

In a hosted VM system, the VMM intervenes in a wider variety of situations. To insure that the OS recognizes the hardware and can install required drivers, the hardware abstraction must replicate standard and familiar hardware for the guest environment and present well-defined emulated interface to the OS. The VMM must then intervene to convert driver outputs intended for the emulated hardware to the requirements of the real hardware. The VMM must also isolate guests from one another and protect itself from guest OS and application software. Guest software must not be allowed to change the allocation of physical (real) resources. Since the VMM runs in kernel mode, the applications and the guest OSs run in user mode — this can create complications for the guest OS. Although virtualization primitives are built into the mainframe ISA, the VMM on a hosted VM must trap any OS or application access to the hardware abstraction layer (HAL).

Virtualization-unaware guest OS is written to run above the physical hardware, and expects full and direct control of real hardware. Correct functioning requires extensive intervention and the VMM must remap (bluff) pointers intended for real hardware.

A virtualization-aware guest OS is written to run above a VMM, and expects to interact with the virtual host. It does not expect full or direct control of physical hardware. The OS code interfaces with the VMM code with no need to remap pointers.

Slide 14 provides a general description of the interventions required by the VMM for standard hardware operations. As shown on slide 15, some systems require full emulation of every guest activity — a common example is running a game designed for one hardware system and machine language on a different system with a different machine language. But most VMMs are intended to run an OS above a different OS written for the same hardware — in this case, most machine instructions can be sent directly to the CPU without VMM emulation.

Slide 16 – 17 Virtualization Difficulties for x86

The IA-32 ISA was designed to provide hardware support to the OS, including, memory segmentation, virtual memory and paging, task management, and interrupt management. IA-32 also defines privilege-based protection for segmentation, paging, interrupts, granting kernel mode rights (ring 0) to the OS and user mode rights (ring 3) to applications.

Initial attempts at virtualization on x86 CPUs attempted a workaround — treating the OS like user application running in user mode above the VMM running in kernel mode. This can create a kludge on these systems. An IA-32 OS expects to have the highest privilege on the system and can easily discover that they run at lower privilege. They also expect to see the full virtual memory space and have direct access to CPU-controlled system tables. Since the VMM must reserve address and table space for itself, the OS suffers memory and table compression.

In its Virtualization Technology (VT) Intel created a new privilege level below kernel mode, called the VMX root. Kernel mode and user mode are called non- VMX root modes. The VMM runs at the VMX root privilege level, so that the OS can run in standard ring 0 kernel mode, and applications run in standard ring 3 user mode. System calls requiring access to I/O or real hardware are trapped by the VMM, which switches to VMX root mode to handle these requests.

Slide 18 – 20 VirtualBox

VirtualBox is an open source hosted VMM by Oracle (acquired when it purchased Sun Microsystems). It runs on Intel and AMD x86 hardware above Windows, Linux, Mac OS X (Intel), and Solaris, providing a VM for a guest OS. It uses hardware virtualization support if available (but is not required). The host OS schedules the VM by granting a timeslice — the guest OS schedules sub-processes in the VM.

VirtualBox is designed as a client/server system. The front-end (client) is the hypervisor that runs above the host OS. The back-end (server) runs as a ring 0 driver in the host OS. In the language of the VirtualBox documentation it copes with the "gory details of x86 architecture," to allocate physical memory for the VM (guest OS), save and restore guest CPU context during a host interrupt, manage IA-32 registers and descriptor tables. It makes no interventions in guest OS process management.

Some code runs native (not emulated) on the CPU: host OS code at ring 0, host applications at ring 3, and "safe" guest application code at ring 3 (non-system activities and code that makes system calls to the guest OS). The code that runs emulated at ring 3 includes guest application code that would disable interrupts or perform prohibited accesses. Naturally, instructions produced by the VirtualBox back-end driver (to replace native I/O operation) run at ring 3. In order to minimize use of VT mode transitions, VirtualBox runs most guest OS ring 0 code at privilege ring 1 (less privileged kernel mode).

Slide 21 – 28 Xen

Xen is an open source system VMM that runs on Intel and AMD x86 hardware directly above the hardware, providing VMs with parallel guest OSs (Linux, Solaris, Windows). Hardware virtualization support is required for a Windows guest OS. For a Linux/Unix guest OS, Xen provides paravirtualization — a modified OS kernel that operates directly with Xen, and a virtual hardware layer very similar to the native hardware.

The Xen hypervisor operates directly above the hardware and boots the system on start-up. The VMM creates one Domain 0 and one or more copies of Domain U. Domain 0 runs XenLinux and Domain Management and Control (DMC). Domain U provides a VM running the guest OS. The hypervisor operates directly above hardware in ring 0 to perform CPU scheduling for the virtual machines, memory partitioning, and provide hardware abstraction to virtual machines. It has no awareness of issues handled by Domain 0: networking, external storage devices, video, or common I/O.

Domain 0 runs XenLinux, a modified Linux kernel running in a unique VM over the hypervisor. It has direct privileged access rights to physical I/O resources and provides I/O virtualization to the Domain U guest VMs. It also provides generic I/O drivers. The Network Backend Driver manages local networking hardware and processes all VM networking requests from Domain U guests. The Block Backend Driver manages local storage disk and processes all read/write data requests from Domain U guests.

A Domain U running Unix or Linux performs paravirtualization, running a modified kernel above a hardware abstraction very similar in structure to the real hardware. This OS expects no direct access to host hardware and knows that it shares host hardware with other VMs. The guest drivers provide I/O access through backend drivers in Domain 0 (the PV Network Driver and PV Block Driver).

A Domain U running a non-virtualization-aware OS is called a Domain U HVM (host VM) and runs fully virtualized machines as VMX non-root operation with Intel VT. The OS expects no hardware virtualization and no sharing with other VMs. It also expects full hardware access for boot. Xen virtual firmware runs as a VMX root operation to simulate the BIOS expected by OS. Domain 0 runs a Qemu-dm daemon for each HVM guest OS to support networking and disk access requests.

Slides 27 – 28 summarize research on the XEN performance. Slide 27 shows that using Intel VT, a Xen system generally performs very well compared to a native OS running on the same hardware. However slide 28 shows another test that revealed an interesting hidden limitation. This experiment shows that the test software experiences a communications bottleneck that limits performance. The native system (without virtualization) experienced an improvement in performance when the number of network cards in the hardware was increased from 1 to 2. The Domain 0 OS in the Xen system experienced a similar performance improvement, but the performance of the Domain U OS did not change (the native system significantly outperformed the virtual system with adequate hardware for the applications). The native system experienced another performance boost when the number of network cards in the hardware was increased to 3, but there was no similar increase for the virtual system. Generally, virtualization can provide many advantages, but does not achieve the performance expected from native systems.

Slide 29 – 33 Cloud Computing

It has become common to use the word **cloud** as a synonym for **network**. But cloud computing refers to a specific mode of operation, essentially a business model rather than a computing model. It is a form of outsourcing in which a customer contracts to receive computing services from a service provider. A service level agreement (SLA) guarantees a level of service to the customer to be enjoyed by the customer's users (usually, the customer's customers). The provider handles the complex details of OAM (operations, administration, maintenance). Cloud computing brings a number of business advantages in appropriate cases, especially the economies of scale possible for large provider. Also, contracting for computing services cuts labor and capital costs from the customer's balance sheet, which can make investors happy. On

the other hand, the contract requires monthly payment (like rent) that can become a major burden on a company struggling for income and cash flow.

Cloud computing is based on standard technologies, not a fundamentally different computing technology. The service is provided using conventional resources — hardware, software, and network — from a menu of services the provider offers customers. The unique technological issues in cloud computing center on service reliability (the provider is committed to the SLA), optimization of the provider-side resource configuration, and optimization of the customer-side resource configuration. In this case, optimization refers both to performance and cost.

Cloud computing is characterized by a service hierarchy:

In **infrastructure as a service** (IaaS), the customer sees a virtual hardware environment, some combination of real hardware and a system virtual machine. The customer installs an OS and application software, and runs jobs.

In **platform as a service (PaaS)**, the customer sees a virtual OS environment, installs application software, and runs jobs.

In **software as a service (SaaS)**, the customer sees a virtual application software environment, and runs jobs. These applications run on a private copy of the OS or are "sandboxed" on shared OS. A sandbox is a private execution environment per application instance that cannot access the resources of other running programs. Google apps is one familiar example of SaaS.

A hybrid category is **storage as a service (STaaS)**, where the user sees a virtual storage device mounted on a local computer. Examples, include Dropbox, Copy, Box, iCloud and others.

The popularity of cloud computing is an example of changing fashions in the computing industry. Slide 31 shows a timeline over the past 60 years, showing the trend away from centralized computing systems to distributed systems, flowed by the return to centralization.

Cost is a major factor in cloud computing. For the provider, economies of scale lower the cost of performing any given compute job. Cost is often the primary factor in competitiveness. For the customer, cloud computing exchanges capital and OAM costs into service costs. If the customer's income does not meet the cost of the cloud service, this exchanges start-up costs for operating debt.

Reliability is a major issue for the provider. The provider's planning must include redundant infrastructure for continuity and disaster recovery, centralized management of OAM, security, performance, for administrative control and cost management. Choices include virtualization, which can provide multiple OSs on one physical server or multitenancy, which providea multiple sandboxed application instances on one OS.

Agility is another advantage of cloud computing. The user or provider can reconfigure the service and infrastructure as needed. This permits smooth growth, effective load balancing, and time-zone serving (where one system is redirected toward a part of the world where demand is greatest, shifting as day turns to night).

In a public cloud, the service provider acts as a public utility, selling (renting) computing service. The initial providers of public cloud service were leveraging their large existing computing infrastructure — Amazon, Microsoft, Google, IBM.

In a private cloud, the cloud infrastructure is managed (internally or outsourced) for a private organization. The goal is to isolate service developers from certain implementation issues. Programmers and lower-level IT personnel see a standard virtual development platform that remains stable over many years, while the real hardware undergoes periodic reorganization. Private cloud computing is economically justified for large organizations offering technology-based services and introducing new services frequently.