**Unit – III**

**Virtualization**

**Cloud deployment models**

•      **Public**: Accessible, via the Internet, to anyone who pays

–     Owned by service providers; e.g., Google App Engine, Amazon Web Services, Force.com.

•      **Community**: Shared by two or more organizations with joint interests, such as colleges within a university

•      **Private**: Accessible via an intranet to the members of the owning organization

–     Can be built using open source software such as CloudStack or OpenStack

–     Example of private cloud: NASA’s cloud for climate modeling

•      **Hybrid**

–     A private cloud might buy computing resources from a public cloud.

**Public, Private, and Hybrid Clouds**

The concept of cloud computing has evolved from cluster, grid, and utility computing. Cluster and grid computing leverage the use of many computers in parallel to solve problems of any size. Utility and Software as a Service (SaaS) provide computing resources as a service with the notion of pay per use. Cloud computing leverages dynamic resources to deliver large numbers of services to end users. Cloud computing is a high-throughput computing (HTC) paradigm whereby the infrastructure provides the services through a large data center or server farms. The cloud computing model enables users to share access to resources from anywhere at any time through their connected devices.

Cloud will free users to focus on user application development and create business value by outsourcing job execution to cloud providers. In this scenario, the computations (programs) are sent to where the data is located, rather than copying the data to millions of desktops as in the traditional approach. Cloud computing avoids large data movement, resulting in much better network bandwidth utilization. Furthermore, machine virtualization has enhanced resource utilization, increased application flexibility, and reduced the total cost of using virtualized data-center resources.

The cloud offers significant benefit to IT companies by freeing them from the low-level task of setting up the hardware (servers) and managing the system software. Cloud computing applies a virtual platform with elastic resources put together by on-demand provisioning of hardware, software, and data sets, dynamically. The main idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers. Cloud computing leverages its low cost and simplicity to both providers and users. According to Ian Foster [25], cloud computing intends to leverage multitasking to achieve higher throughput by serving many heterogeneous applications, large or small, simultaneously.

**Centralized versus Distributed Computing**

Some people argue that cloud computing is centralized computing at data centers. Others claim that cloud computing is the practice of distributed parallel computing over data-center resources. These represent two opposite views of cloud computing. All computations in cloud applications are distributed to servers in a data center. These are mainly virtual machines (VMs) in virtual clusters created out of data-center resources. In this sense, cloud platforms are systems distributed through virtualization. As Figure 4.1 shows, both public clouds and private clouds are developed in the Internet. As many clouds are generated by commercial providers or by enterprises in a distributed manner, they will be interconnected over the Internet to achieve scalable and efficient computing services. Commercial cloud providers such as Amazon, Google, and Microsoft created their platforms to be distributed geographically.

This distribution is partially attributed to fault tolerance, response latency reduction, and even legal reasons. Intranet-based private clouds are linked to public clouds to get additional resources. Nevertheless, users in Europe may not feel comfortable using clouds in the United States, and vice versa, until extensive service-level agreements (SLAs) are developed between the two user communities.

**Public Clouds**

A public cloud is built over the Internet and can be accessed by any user who has paid for the service. Public clouds are owned by service providers and are accessible through a subscription. The callout box in top of Figure 4.1 shows the architecture of a typical public cloud. Many public clouds are available, including Google App Engine (GAE), Amazon Web Services (AWS), Microsoft Azure, IBM Blue Cloud, and Salesforce.com’s Force.com. The providers of the aforementioned clouds are commercial providers that offer a publicly accessible remote interface for creating and managing VM instances within their proprietary infrastructure. A public cloud delivers a selected set of business processes. The application and infrastructure services are offered on a flexible price-per-use basis.

**Private Clouds**

A private cloud is built within the domain of an intranet owned by a single organization. Therefore,it is client owned and managed, and its access is limited to the owning clients and their partners. Itsdeployment was not meant to sell capacity over the Internet through publicly accessible interfaces.

****

Private clouds give local users a flexible and agile private infrastructure to run service workloads within their administrative domains. A private cloud is supposed to deliver more efficient and convenient cloud services. It may impact the cloud standardization, while retaining greater customization and organizational control.

**Hybrid Clouds**

A hybrid cloud is built with both public and private clouds, as shown at the lower-left corner of Figure 4.1. Private clouds can also support a hybrid cloud model by supplementing local infrastructure with computing capacity from an external public cloud. For example, the Research Compute Cloud (RC2) is a private cloud, built by IBM, that interconnects the computing and IT resources at eight IBM Research Centers scattered throughout the United States, Europe, and Asia. A hybrid cloud provides access to clients, the partner network, and third parties. In summary, public clouds promote standardization, preserve capital investment, and offer application flexibility. Private clouds attempt to achieve customization and offer higher efficiency, resiliency, security, and privacy. Hybrid clouds operate in the middle, with many compromises in terms of resource sharing.

**Data-Center Networking Structure**

The core of a cloud is the server cluster (or VM cluster). Cluster nodes are used as compute nodes. A few control nodes are used to manage and monitor cloud activities. The scheduling of user jobs requires that you assign work to virtual clusters created for users. The gateway nodes provide the access points of the service from the outside world. These gateway nodes can be also used for security control of the entire cloud platform. In physical clusters and traditional grids, users expect static demand of resources. Clouds are designed to handle fluctuating workloads, and thus demand variable resources dynamically. Private clouds will satisfy this demand if properly designed and managed.

Data centers and supercomputers have some similarities as well as fundamental differences. In the case of data centers, scaling is a fundamental requirement. Data-center server clusters are typically built with large number of servers, ranging from thousands to millions of servers (nodes). For example, Microsoft has a data center in the Chicago area that has 100,000 eight-core servers, housed in 50 containers. In supercomputers, a separate data farm is used, while a data center uses disks on server nodes plus memory cache and databases.

Data centers and supercomputers also differ in networking requirements. Supercomputers use custom-designed high-bandwidth networks such as fat trees or 3D torus networks (which we discussed in Chapter 2). Data-center networks are mostly IP-based commodity networks, such as the 10 Gbps Ethernet network, which is optimized for Internet access. Figure 4.2 shows a multilayer structure for accessing the Internet. The server racks are at the bottom Layer 2, and they are connected through fast switches (S) as the hardware core. The data center is connected to the Internet at Layer 3 with many access routers (ARs) and border routers (BRs).



An example of a private cloud is the one the U.S. National Aeronautics and Space Administration (NASA) is building to enable researchers to run climate models on remote systems it provides. This can save users the capital expense of HPC machines at local sites. Furthermore, NASA can build the complex weather models around its data centers, which is more cost-effective. Another good example is the cloud built by the European Council for Nuclear Research (CERN). This is a very big private cloud designed to distribute data, applications, and computing resources to thousands of scientists around the world.

These cloud models demand different levels of performance, data protection, and security enforcement. In this case, different SLAs may be applied to satisfy both providers and paid users. Cloud computing exploits many existing technologies. For example, grid computing is the backbone of cloud computing in that the grid has the same goals of resource sharing with better utilization of research facilities. Grids are more focused on delivering storage and computing resources while cloud computing aims to achieve economies of scale with abstracted services and resources.

**3 service models**

•      Cloud Software as a Service (SaaS)

–     Use provider’s applications over a network

•      Cloud Platform as a Service (PaaS)

–     Deploy customer-created applications to a cloud

•      Cloud Infrastructure as a Service (IaaS)

–     Rent processing, storage, network capacity, and other fundamental computing resources

**5 essential characteristics**

•      On-demand self-service: consumers can acquire the necessary computational resources without having to interact with human service providers.

•      Ubiquitous network access: cloud features don’t require special devices – laptops, mobile phones, etc. are generally supported.

•      Resource pooling: cloud resources are pooled to serve many customers “… using a multi-tenant model, with different physical and virtual resources…”

•      Rapid elasticity: resources can be allocated and de-allocated quickly as needed.

•      Measured service: resource use is measured and monitored; charges are made based on usage and service type (e.g., storage, CPU cycles, etc.)

**Infrastructure as a Service**

This model allows users to use virtualized IT resources for computing, storage, and networking. Inshort, the service is performed by rented cloud infrastructure. The user can deploy and run his applications over his chosen OS environment. The user does not manage or control the underlying cloud infrastructure, but has control over the OS, storage, deployed applications, and possibly select networking components. This IaaS model encompasses storage as a service, compute instances as a service, and communication as a service. The Virtual Private Cloud (VPC) in Example 4.1 shows how to provide Amazon EC2 clusters and S3 storage to multiple users. Many startup cloud providers have appeared in recent years. GoGrid, FlexiScale, and Aneka are good examples. Table 4.1 summarizes the IaaS offerings by five public cloud providers. Interested readers can visit the companies’ web sites for updated information.



**Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS)**

In this section, we will introduce the PaaS and SaaS models for cloud computing. SaaS is often built on top of the PaaS, which is in turn built on top of the IaaS.

**Platform as a Service (PaaS)**

To be able to develop, deploy, and manage the execution of applications using provisioned resources demands a cloud platform with the proper software environment. Such a platform includes operating system and runtime library support. This has triggered the creation of the PaaS model to enable users to develop and deploy their user applications. The platform cloud is an integrated computer system consisting of both hardware and software infrastructure. The user application can be developed on this virtualized cloud platform using some programming languages and software tools supported by the provider (e.g., Java, Python, .NET). The user does not manage the underlying cloud infrastructure. The cloud provider supports user application development and testing on a well-defined service platform. This PaaS model enables a collaborated software development platform for users from different parts of the world. This model also encourages third parties to provide software management, integration, and service monitoring solutions.

Five Public Cloud Offerings of PaaS [10,18]

|  |  |  |  |
| --- | --- | --- | --- |
| Cloud Name | Languages and Developer Tools | Programming Models Supported by Provider | Target Applications and Storage Option |
| Google App Engine | Python, Java, and Eclipse-based IDE | MapReduce, web programming on demand | Web applications andBigTable storage |
| Salesforce.com’s Force.com | Apex, Eclipse-based IDE, web-based Wizard | Workflow, Excel-like formula, Web programming on demand | Business applications such as CRM |
| Microsoft Azure | .NET, Azure tools for MS Visual Studio | Unrestricted model | Enterprise and webapplications |
| Amazon ElasticMapReduce | Hive, Pig, Cascading,Java, Ruby, Perl,Python, PHP, R, C++ | MapReduce | Data processing ande-commerce |
| Aneka | .NET, stand-alone SDK | Threads, task, MapReduce | .NET enterpriseapplications, HPC |



***Cloud Design Objectives***

The following list highlights six design objectives for cloud computing:

• **Shifting computing from desktops to data centers**Computer processing, storage, and software delivery is shifted away from desktops and local servers and toward data centers over the Internet.

• **Service provisioning and cloud economics**Providers supply cloud services by signing SLAs with consumers and end users. The services must be efficient in terms of

computing, storage, and power consumption. Pricing is based on a pay-as-you-go policy.

• **Scalability in performance**the cloud platforms and software and infrastructure services must be able to scale in performance as the number of users increases.

• **Data privacy protection**Can you trust data centers to handle your private data and

Records? This concern must be addressed to make clouds successful as trusted services.

• **High quality of cloud services**The QoS of cloud computing must be standardized to make clouds interoperable among multiple providers.

• **New standards and interfaces**this refers to solving the data lock-in problem associated with data centers or cloud providers. Universally accepted APIs and access

Protocols are needed to provide high portability and flexibility of virtualized applications.

**Software as a Service (SaaS)**

            This refers to browser-initiated application software over thousands of cloud customers. Services and tools offered by PaaS are utilized in construction of applications and management of their deployment on resources offered by IaaS providers. The SaaS model provides software applications as a service. As a result, on the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are kept rather low, compared with conventional hosting of user applications. Customer data is stored in the cloud that is either vendor proprietary or publicly hosted to support PaaS and IaaS. The best examples of SaaS services include Google Gmail and docs, Microsoft SharePoint, and the CRM software from Salesforce.com. They are all very successful in promoting their own business or are used by thousands of small businesses in their dayto- day operations. Providers such as Google and Microsoft offer integrated IaaS and PaaS services, whereas others such as Amazon and GoGrid offer pure IaaS services and expect third-party PaaS providers such as Manjrasoft to offer application development and deployment services on top of their infrastructure services.

**Implementation levels of virtualization**

            A traditional computer runs with a host operating system specially tailored for its hardware architecture. After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This is often done by adding additional software, called a virtualization layer. This virtualization layer is known as hypervisor or virtual machine monitor (VMM). The VMs are shown in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources.

The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. This can be implemented at various operational levels, as we will discuss shortly. The

Virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system. Common virtualization layers include the

*Instruction set architecture (ISA)*level, hardware level, operating system level, and library support level, and application level.



Figure: Virtualization ranging from hardware to applications in five abstraction levels

**Instruction Set Architecture Level**

            At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation. The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one. One source instruction may require tens or hundreds of native target instructions to perform its function. This approach translates basic blocks of dynamic source instructions to target instructions. The basic blocks can also be extended to program traces or super blocks to increase translation efficiency. Instruction set emulation requires binary translation and optimization. A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

**Hardware Abstraction Level**

            Hardware-level virtualization is performed right on top of the bare hardware. On the one hand, this approach generates a virtual hardware environment for a VM. On the other hand, the process manages the underlying hardware through virtualization. The idea is to virtualize a computer’s resources, such as its processors, memory, and I/O devices. The intention is to upgrade the hardware utilization rate by multiple users concurrently. The idea was implemented in the IBM VM/370 in the 1960s. More recently, the Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS.

**Operating System Level**

            This refers to an abstraction layer between traditional OS and user applications. OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers. The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.

**Library Support Level**

            Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS. Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization. Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts.

**User-Application Level**

            Virtualization at the application level virtualizes an application as a VM. On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy high level language (HLL) VMs. In this scenario, the virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. Any program written in the HLL and compiled for this VM will be able to run on it. The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM.

**Relative Merits of Different Approaches**

The column headings correspond to four technical merits. “Higher Performance” and “Application Flexibility” are self-explanatory. “Implementation Complexity” implies the cost to implement that particular virtualization level. “Application Isolation” refers to the effort required to isolate resources committed to different VMs. Each row corresponds to a particular level of virtualization.



**Virtualization of CPU, Memory, and I/O Devices?**

**Hardware Support for Virtualization**

            Modern operating systems and processors permit multiple processes to run simultaneously. If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash. Therefore, all processors have at least two modes, user mode and supervisor mode, to ensure controlled access of critical hardware. Instructions running in supervisor mode are called privileged instructions. Other instructions are unprivileged instructions. The

VMware Workstation is a VM software suite for x86 and x86-64 computers. This software suite allows users to set up multiple x86 and x86-64 virtual computers and to use one or more of these VMs simultaneously with the host operating system. The VMware Workstation assumes the host-based virtualization. Xen is a hypervisor for use in IA-32, x86-64, Itanium, and PowerPC 970 hosts.

**CPU Virtualization**

            A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode. Thus, unprivileged instructions of VMs run directly on the host machine for higher efficiency. Other critical instructions should be handled carefully for correctness and stability. The critical instructions are divided into three categories: privileged instructions, control-sensitive instructions, and behavior-sensitive instructions. Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode. Control-sensitive instructions attempt to change the configuration of resources used. Behavior-sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

The VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system.

 **Hardware-Assisted CPU Virtualization**

                        This technique attempts to simplify virtualization because full or paravirtualization is complicated. Intel and AMD add an additional mode called privilege mode level to x86 processors.

**Memory Virtualization**

            Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance. A two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory. The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. The MMU already handles virtual-to-physical translations as defined by the OS. Then the physical memory addresses are translated to machine addresses using another set of page tables defined by the hypervisor. Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access. When the guest OS changes the virtual memory to a physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup.

**I/O Virtualization**

            I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware. At the time of this writing, there are three ways to implement I/O virtualization: full device emulation, para-virtualization, and direct I/O.

            Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well-known, real-world devices. All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices.

            The para-virtualization method of I/O virtualization is typically used in Xen. It is also known as the split driver model consisting of a frontend driver and a backend driver. The frontend driver is running in Domain U and the backend driver is running in Domain 0.They interact with each other via a block of shared memory. The frontend driver manages the I/O requests of the guest OSes and the backend driver is responsible for managing the real I/O devices and multiplexing the I/O data of different VMs.

**Virtualization in Multi-Core Processors**

                        Virtualizing a multi-core processor is relatively more complicated than virtualizing a unicore processor. Though multicore processors are claimed to have higher performance by integrating multiple processor cores in a single chip, muti-core virtualization has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers. There are mainly two difficulties: Application programs must be parallelized to use all cores fully, and software must explicitly assign tasks to the cores, which is a very complex problem.

**Physical versus Virtual Processor Cores**

                                    This technique alleviates the burden and inefficiency of managing hardware resources by software. It is located under the ISA and remains unmodified by the operating system or VMM (hypervisor).

**Virtual Hierarchy**

    The emerging many-core chip multiprocessors (CMPs) provides a new computing landscape. To optimize for space-shared workloads, they propose using virtual hierarchies to overlay a coherence and caching hierarchy onto a physical processor. Unlike a fixed physical hierarchy, a virtual hierarchy can adapt to fit how the work is space shared for improved performance and performance isolation.

            In a cluster built with mixed nodes of host and guest systems, the normal method of operation is to run everything on the physical machine. When a VM fails, its role could be replaced by another VM on a different node, as long as they both run with the same guest OS. The advantage is enhanced failover flexibility.

The migration copies the VM state file from the storage area to the host machine.

There are four ways to manage a virtual cluster.

            First, you can use a guest-based manager, by which the cluster manager resides on a guest system. The host-based manager supervises the guest systems and can restart the guest system on another physical machine.

            These two cluster management systems are either guest-only or host-only, but they do not mix.

            A third way to manage a virtual cluster is to use an independent cluster manager on both the host and guest systems. This will make infrastructure management more complex.

            Finally, you can use an integrated cluster on the guest and host systems.

VM can be in one of the following four states.

            An **inactive** state is defined by the virtualization platform, under which the VM is not enabled.

            An **active** state refers to a VM that has been instantiated at the virtualization platform to perform a real task.

            A **paused** state corresponds to a VM that has been instantiated but disabled to process a task or paused in a **waiting** state.

Live migration of a VM consists of the following six steps:

Steps 0 and 1: **Start migration**. This step makes preparations for the migration, including determining the migrating VM and the destination host. Although users could manually make a VM migrate to an appointed host, in most circumstances, the migration is automatically started by strategies such as load balancing and server consolidation.

Steps 2: **Transfer memory**. Since the whole execution state of the VM is stored in memory, sending the VM’s memory to the destination node ensures continuity of the service provided by the VM. All of the memory data is transferred in the first round, and then the migration controller recopies the memory data which is changed in the last round. These steps keep iterating until the dirty portion of the memory is small enough to handle the final copy. Although precopying memory is performed iteratively, the execution of programs is not obviously interrupted.

Step 3: **Suspend** the VM and copy the last portion of the data. The migrating VM’s execution is suspended when the last round’s memory data is transferred. Other nonmemory data such as CPU and network states should be sent as well. During this step, the VM is stopped and its applications will no longer run. This “service unavailable” time is called the “downtime” of migration, which should be as short as possible so that it can be negligible to users.

Steps 4 and 5: **Commit and activate** the new host. After all the needed data is copied, on the destination host, the VM reloads the states and recovers the execution of programs in it, and the service provided by this VM continues. Then the network connection is redirected to the new VM and the dependency to the source host is cleared. The whole migration process finishes by removing the original VM from the source host.

Virtualization for data center automation

            A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN.

            Virtual clusters are built with VMs installed at distributed servers from one or more physical clusters. The VMs in a virtual cluster are interconnected logically by a virtual network across several physical networks. Each virtual cluster is formed with physical machines or a VM hosted by multiple physical clusters. The virtual cluster boundaries are shown as distinct boundaries.



The provisioning of VMs to a virtual cluster is done dynamically to have the following

Interesting properties:

• The virtual cluster nodes can be either physical or virtual machines. Multiple VMs running with different OSes can be deployed on the same physical node.

• A VM runs with a guest OS, which is often different from the host OS, that manages the resources in the physical machine, where the VM is implemented.

• The purpose of using VMs is to consolidate multiple functionalities on the same server. This will greatly enhance server utilization and application flexibility.

• VMs can be colonized (replicated) in multiple servers for the purpose of promoting distributed parallelism, fault tolerance, and disaster recovery.

• The size (number of nodes) of a virtual cluster can grow or shrink dynamically, similar to the way an overlay network varies in size in a peer-to-peer (P2P) network.

• The failure of any physical nodes may disable some VMs installed on the failing nodes. But the failure of VMs will not pull down the host system.



Figure: A Virtual Clusters based on Application Partitioning:

Parallax Providing Virtual Disks to Clients VMs from a Large Common Shared Physical Disk.



Cloud OS for Building Private Clouds (VI: Virtual Infrastructure, EC2: Elastic Compute Cloud).



Eucalyptus: An Open-Source OS for Setting Up and Managing Private Clouds (IaaS)

Three Resource Managers: CM (Cloud Manager), GM (Group Manager), and IM (Instance Manager)Works like AWS APIs



VMware vSphere 4 – A Commercial Cloud OS.



VM-based Intrusion Detection.



Techniques for establishing trusted zones for virtual cluster insulation and VM isolation