UNIT V VIRTUAL MACHINES AND MOBILE OS

Virtual Machines – History, Benefits and Features, Building Blocks, Types of Virtual Machines and their Implementations, Virtualization and Operating-System Components; Mobile OS - iOS and Android.

I Virtual Machines

Virtual Machine abstracts the hardware of our personal computer such as CPU, disk drives, memory, NIC (Network Interface Card) etc, into many different execution environments as per our requirements, hence giving us a feel that each execution environment is a single computer. For example, VirtualBox.

When we run different processes on an operating system, it creates an illusion that each process is running on a different processor having its own virtual memory, with the help of CPU scheduling and virtual-memory techniques. There are additional features of a process that cannot be provided by the hardware alone like system calls and a file system. The virtual machine approach does not provide these additional functionalities but it only provides an interface that is same as basic hardware. Each process is provided with a virtual copy of the underlying computer system.

We can create a virtual machine for several reasons, all of which are fundamentally related to the ability to share the same basic hardware yet can also support different execution environments, i.e., different operating systems simultaneously.

Advantages:

1. There are no protection problems because each virtual machine is completely isolated from all other virtual machines.
2. Virtual machine can provide an instruction set architecture that differs from real computers.
3. Easy maintenance, availability and convenient recovery.

Disadvantages:

1. When multiple virtual machines are simultaneously running on a host computer, one virtual machine can be affected by other running virtual machines, depending on the workload.
2. Virtual machines are not as efficient as a real one when accessing the hardware.

HISTORY

Both system virtual machines and process virtual machines date to the 1960s and continue to be areas of active development.

System virtual machines grew out of time-sharing, as notably implemented in the Compatible Time-Sharing System (CTSS). Time-sharing allowed multiple users to use a computer concurrently: each program appeared to have full access to the machine, but only one program was executed at the time, with the system switching between programs in time slices, saving

and restoring state each time.

Process virtual machines arose originally as abstract platforms for an intermediate language used as the intermediate representation of a program by a compiler; early examples date to around 1966. An early 1966 example was the O-code machine, a virtual machine that executes O-code (object code) emitted by the front end of the BCPL compiler. This abstraction allowed the compiler to be easily ported to a new architecture by implementing a new back end that took the existing O-code and compiled it to machine code for the underlying physical machine.

This has been influential, and virtual machines in this sense have been often generally called p-code machines. In addition to being an intermediate language, Pascal p-code was also executed directly by an interpreter implementing the virtual machine, notably in UCSD Pascal (1978); this influenced later interpreters, notably the Java virtual machine (JVM).

Macros have since fallen out of favor, however, so this approach has been less influential. Process virtual machines were a popular approach to implementing early microcomputer software, including Tiny BASIC and adventure games, from one-off implementations such as Pyramid 2000 to a general-purpose engine like Infocom's z-machine, which Graham Nelson argues is "possibly the most portable virtual machine ever created".

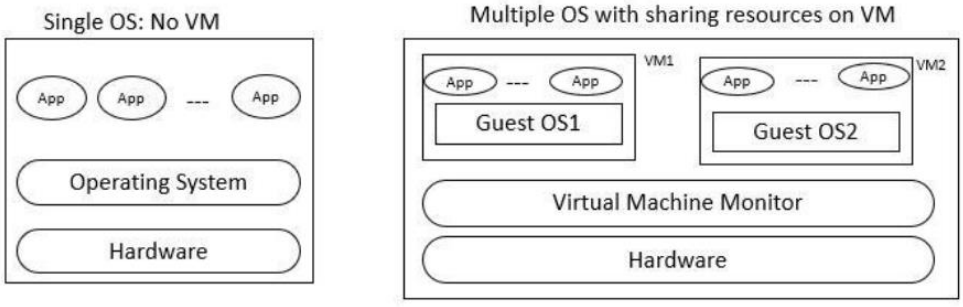
Significant advances occurred in the implementation of Smalltalk-80, particularly the Deutsch/Schiffmann implementation which pushed just-in-time (JIT) compilation forward as an implementation approach that uses process virtual machine. Later notable Smalltalk VMs were VisualWorks, the Squeak Virtual Machine, and Strongtalk.

FEATURES OF VIRTUAL MACHINES

The features of the virtual machines are as follows −

- Multiple OS systems use the same hardware and partition resources between virtual computers.
- Separate Security and configuration identity.
- Ability to move the virtual computers between the physical host computers as holistically integrated files.

The below diagram shows you the difference between the single OS with no VM and Multiple OS with VM −



BENEFITS

Let us see the major benefits of virtual machines for operating-system designers and users which are as follows −

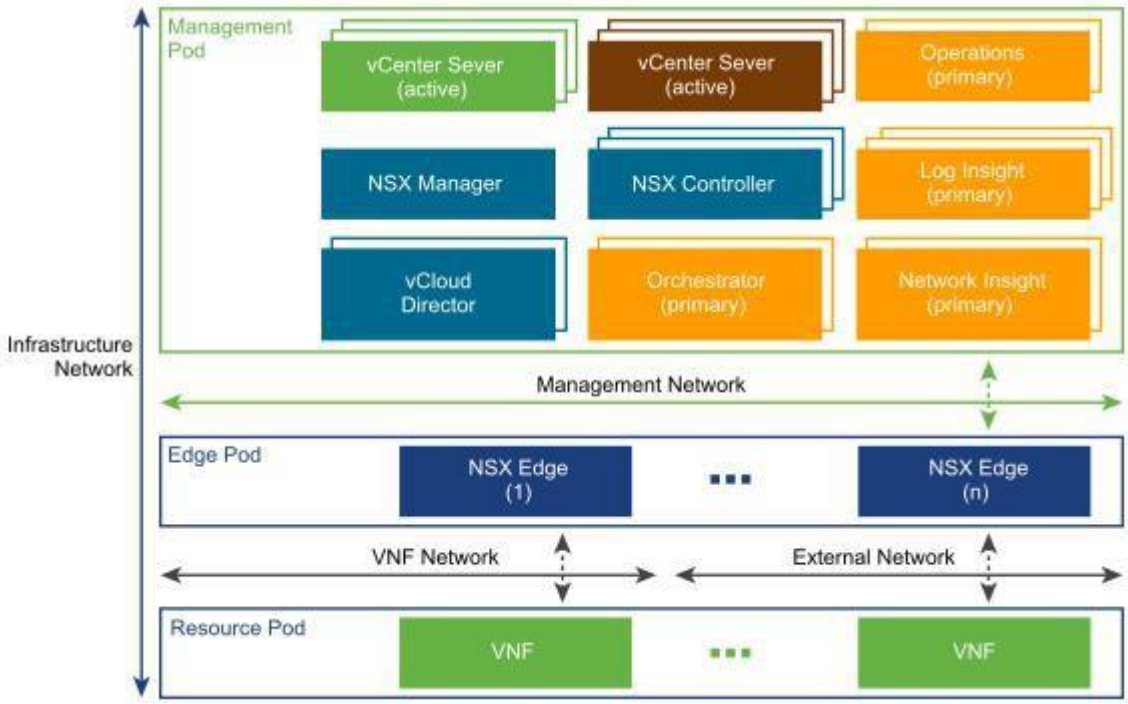
- The multiple Operating system environments exist simultaneously on the same machine, which is isolated from each other.
- Virtual machine offers an instruction set architecture which differs from real computer.
- Using virtual machines, there is easy maintenance, application provisioning, availability and convenient recovery.

Virtual Machine encourages the users to go beyond the limitations of hardware to achieve their goals.

The operating system achieves virtualization with the help of a specialized software called a hypervisor, which emulates the PC client or server CPU, memory, hard disk, network and other hardware resources completely, enabling virtual machines to share resources.

The hypervisor can emulate multiple virtual hardware platforms that are isolated from each other allowing virtual machines to run Linux and window server operating machines on the same underlying physical host.

VIRTUAL BUILDING BLOCKS



Storage Design

This reference architecture uses a shared storage design that is based on vSAN. vCloud NFV also supports certified third-party shared storage solutions, as listed in the VMware Compatibility Guide.

vSAN is a software feature built in the ESXi hypervisor that allows locally attached storage to be pooled and presented as a shared storage pool for all hosts in a vSphere cluster. This simplifies the storage configuration with a single datastore per cluster for management and VNF workloads. With vSAN, VM data is stored as objects and components. One object

consists of multiple components, which are distributed across the vSAN cluster based on the policy that is assigned to the object. The policy for the object ensures a highly available storage backend for the cluster workload, with no single point of failure.

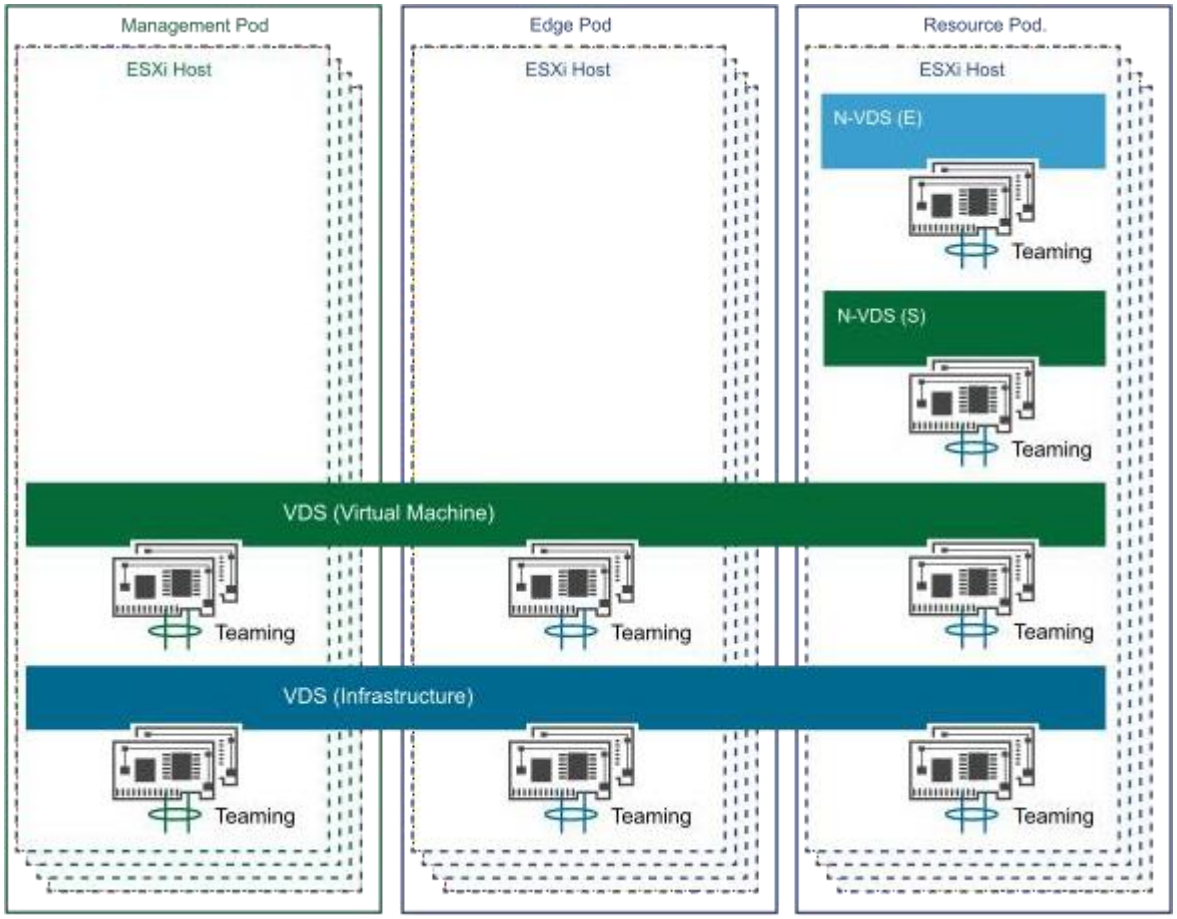
vSAN is a fully integrated hyperconverged storage software. Creating a cluster of server hard disk drives (HDDs) and solid-state drives (SSDs), vSAN presents a flash-optimized, highly resilient, shared storage datastore to ESXi hosts and virtual machines. This allows for the control of capacity, performance, and availability through storage policies, on a per VM basis.

Network Design

Thev Cloud NFV platform consists of infrastructure networks and VM networks. Infrastructure networks are host level networks that connect hypervisors to physical networks. Each ESXi host has multiple port groups configured for each infrastructure network.

The hosts in each Pod are configured with VMware vSphere® Distributed Switch™ (VDS) devices that provide consistent network configuration across multiple hosts. One vSphere Distributed Switch is used for VM networks and the other one maintains the infrastructure networks. Also, the N-VDS switch is used as the transport for telco workload traffic.

Virtual Network Design



Infrastructure networks are used by the ESXi hypervisor for vMotion, VMware vSphere Replication, vSAN traffic, and management and backup. The Virtual Machine networks are

used by VMs to communicate with each other. For each Pod, the separation between infrastructure and VM networks ensures security and provides network resources where needed. This separation is implemented by two vSphere Distributed Switches, one for infrastructure networks and another one for VM networks. Each distributed switch has separate uplink connectivity to the physical data center network, completely separating its traffic from other network traffic. The uplinks are mapped to a pair of physical NICs on each ESXi host, for optimal performance and resiliency.

VMs can be connected to each other over a VLAN or over Geneve-based overlay tunnels. Both networks are designed according to the requirements of the workloads that are hosted by a specific Pod. The infrastructure vSphere Distributed Switch and networks remain the same regardless of the Pod function. However, the VM networks depend on the networks that the specific Pod requires. The VM networks are created by NSX-T Data Center to provide enhanced networking services and performance to the Pod workloads. The ESXi host's physical NICs are used as uplinks to connect the distributed switches to the physical network switches. All ESXi physical NICs connect to layer 2 or layer 3 managed switches on the physical network. It is common to use two switches for connecting to the host physical NICs for redundancy purposes.