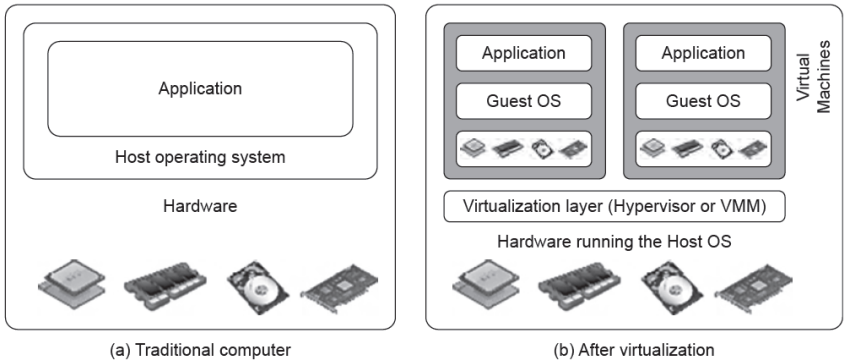
Implementation Levels of Virtualization

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility. The idea is to separate the hardware from the software to yield better system efficiency.

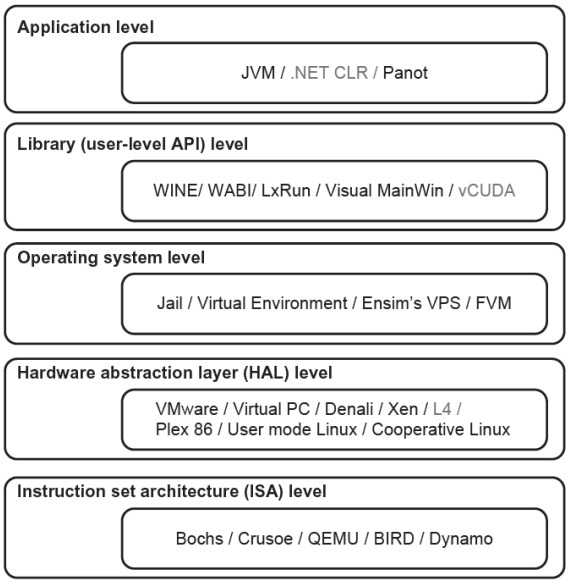
A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure. After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This is often done by adding additional software, called a virtualization layer as shown in Figure.



(a) Traditional computer (b) After virtualization

Virtualization can be implemented at various operational levels, as given below:

- Instruction set architecture (ISA) level
- Hardware level
- Operating system level
- Library support level
- Application level



Instruction Set Architecture Level

At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. 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. Obviously, this process is relatively slow. For better performance, dynamic binary translation is desired. 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. A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

Hardware Abstraction Level

It is performed right on top of the bare hardware and 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 so as hardware utilization rate by multiple users concurrently may be upgraded

Operating System Level

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

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. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

User-Application Level

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 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. Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications. The result is an application that is much easier to distribute and remove from user workstations.

Virtualization Support at the OS Level

It is slow to initialize a hardware-level VM because each VM creates its own image from scratch and storage of such images are also slow. OS-level virtualization provides a feasible solution for these hardware-level virtualization issues. OS virtualization inserts a virtualization layer inside an operating system to partition a machine's physical resources. It enables multiple isolated VMs within a single operating system kernel. This kind of VM is often called a virtual execution environment (VE). This VE has its own set of processes, file system, user accounts, network interfaces with IP addresses, routing tables, firewall rules, and other personal settings.

Advantages:

- VMs at the operating system level have minimal startup/shutdown costs, low resource requirements, and high Scalability
- It is possible for a VM and its host environment to synchronize state changes when necessary

Virtualization Structures/Tools and Mechanisms

Before virtualization, the operating system manages the hardware. After virtualization, a virtualization layer is inserted between the hardware and the OS. In such a case, the virtualization layer is responsible for converting portions of the real hardware into virtual hardware. Depending on the position of the virtualization layer, there are several classes of VM architectures, namely

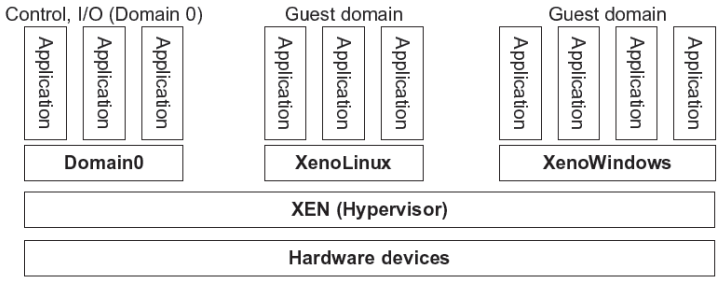
- Hypervisor architecture,
- Paravirtualization
- host-based virtualization.

Hypervisor and Xen Architecture

Depending on the functionality, a hypervisor can assume a micro-kernel architecture or a monolithic hypervisor architecture. A micro-kernel hypervisor includes only the basic and unchanging functions (such as physical memory management and processor scheduling). The device drivers and other changeable components are outside the hypervisor. A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers. Therefore, the size of the hypervisor code of a micro-kernel hypervisor is smaller than that of a monolithic hypervisor.

Xen Architecture

Xen is an open source hypervisor program developed by Cambridge University. Xen is a microkernel hypervisor, which separates the policy from the mechanism. It implementsall the mechanisms, leaving the policy to be handled by Domain 0, as shown in Figure. Xen does not include any device drivers natively. It just provides a mechanism by which a guest OS can have direct access to the physical devices.



Like other virtualization systems, many guest OSes can run on top of the hypervisor. The guest OS (privileged guest OS), which has control ability, is called Domain 0, and the others are called Domain U. It is first loaded when Xen boots without any file system drivers being available. Domain 0 is designed to access hardware directly and manage devices.

Binary Translation with Full Virtualization

Depending on implementation technologies, hardware virtualization can be classified into two categories: full virtualization and host-based virtualization.

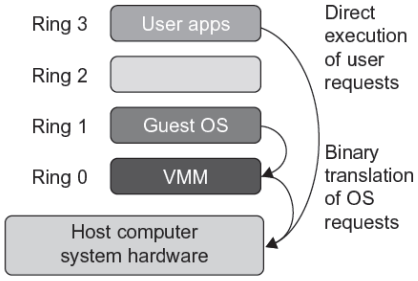
Full Virtualization

With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software.

Both the hypervisor and VMM approaches are considered full virtualization. Noncritical instructions do not control hardware or threaten the security of the system, but critical instructions do. Therefore, running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

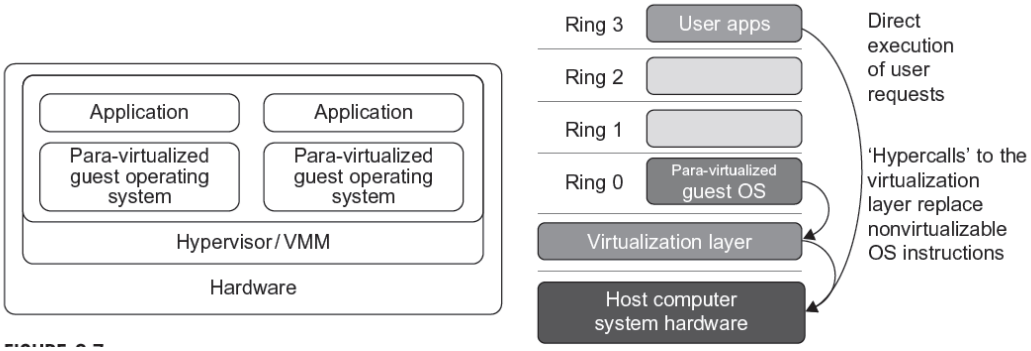
Host-Based Virtualization

An alternative VM architecture is to install a virtualization layer on top of the host OS. This host OS is still responsible for managing the hardware. The guest OSes are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs. Certainly, some other applications can also run with the host OS directly. This host based architecture has some distinct advantages, as enumerated next. First, the user can install this VM architecture without modifying the host OS. Second, the host-based approach appeals to many host machine configurations.



Para-Virtualization

It needs to modify the guest operating systems. A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. Performance degradation is a critical issue of a virtualized system. Figure illustrates the concept of a para-virtualized VM architecture. The guest OS are para-virtualized. They are assisted by an intelligent compiler to replace the non virtualizable OS instructions by hypercalls. The traditional x86 processor offers four instruction execution rings: Rings 0, 1, 2, and 3. The lower the ring number, the higher the privilege of instruction being executed. The OS is responsible for managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3.



Although para-virtualization reduces the overhead, it has incurred problems like compatibility and portability, because it must support the unmodified OS as well. Second, the cost is high, because they may require deep OS kernel modifications. Finally, the performance advantage of para-virtualization varies greatly due to workload variations.

Virtualization of CPU, Memory, And I/O Devices

To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization. In this way, the VMM and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the VMM. To save processor states, mode switching is completed by hardware.

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. In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack. Figure shows the hardware support by Intel.

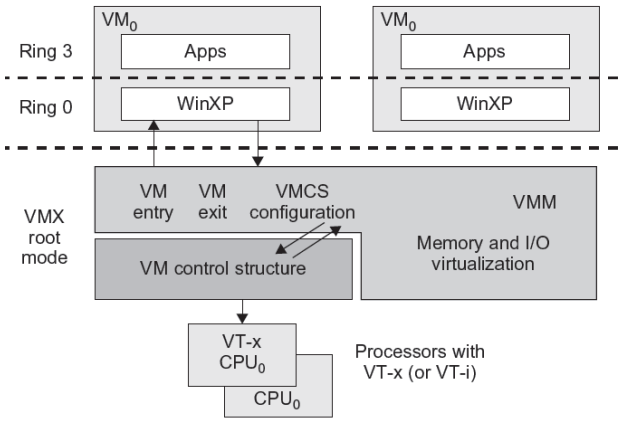
CPU Virtualization

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, controls 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.

A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode. When the privileged instructions including control- and behavior-sensitive instructions of a VM are executed, they are trapped in the VMM. RISC CPU architectures can be naturally virtualized because all control- and behavior-sensitive instructions are privileged instructions. On the contrary, x86 CPU architectures are not primarily designed to support virtualization.

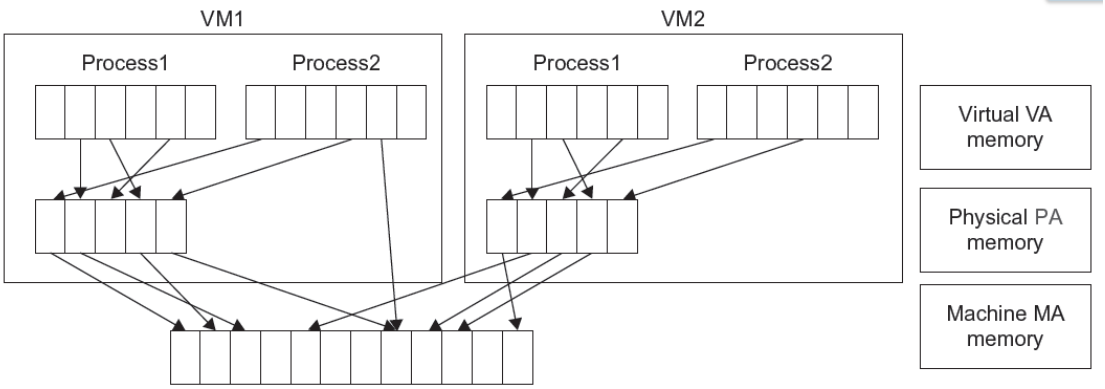
Hardware-Assisted CPU Virtualization

This technique attempts to simplify virtualization because full or paravirtualization is complicated. Intel and AMD add an additional mode called privilege mode level (some people call it Ring-1) to x86 processors. Therefore, operating systems can still run at Ring 0 and the hypervisor can run at Ring -1. All the privileged and sensitive instructions are trapped in the hypervisor automatically. This technique removes the difficulty of implementing binary translation of full virtualization. It also lets the operating system run in VMs without modification



Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional environment, the OS maintains page table for mappings of virtual memory to machine memory, which is a one-stage mapping. All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance. However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs. 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 VMM is responsible for mapping the guest physical memory to the actual machine memory in guest OS.



Since each page table of the guest OSes has a separate page table in the VMM corresponding to it, the VMM page table is called the shadow page table. VMware uses shadow page tables to perform virtual-memory-to-machine-memory address translation. 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

It involves managing the routing of I/O requests between virtual devices and the shared physical hardware. There are three ways to implement I/O virtualization:

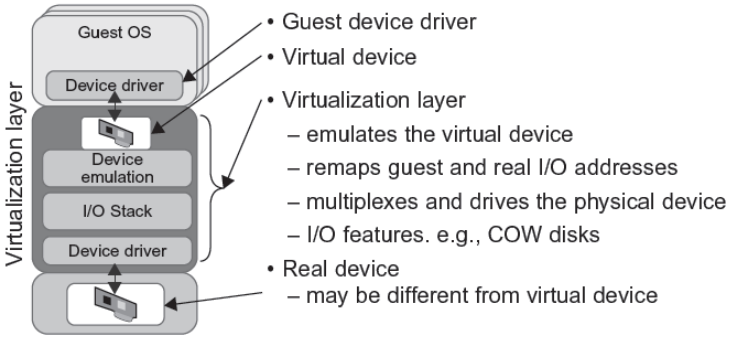
Full device emulation

Para-virtualization

Direct I/O.

Full device emulation

All the functions of a device like device enumeration, identification, interrupts, and DMA, are replicated in software and it 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.



Para-virtualization

It is a 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. Although para-I/O-virtualization achieves better device performance than full device emulation, it comes with a higher CPU overhead.

Direct I/O virtualization

It lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs. However, current direct I/O virtualization implementations focus on networking for mainframes.

Another way to help I/O virtualization is via self-virtualized I/O (SV-IO). The key idea is to harness the rich resources of a multicore processor. All tasks associated with virtualizing an I/O device are encapsulated in SV-IO. SV-IO defines one virtual interface (VIF) for every kind of virtualized I/O device, such as virtual network interfaces, virtual block devices (disk), virtual camera devices, and others. The guest OS interacts with the VIFs via VIF device drivers. Each VIF consists of two message queues. One is for outgoing messages to the devices and the other is for incoming messages from the devices. In addition, each VIF has a unique ID for identifying it in SV-IO.

Virtualization in Multi-Core Processors

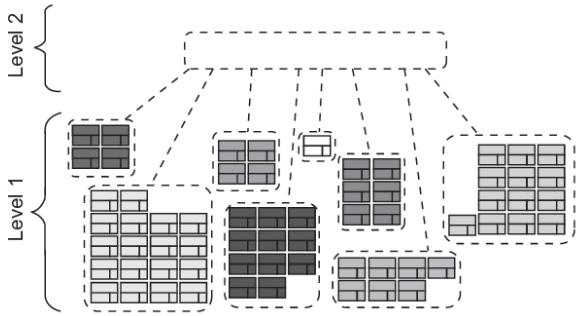
Multicore processors are claimed to have higher performance by integrating multiple processor cores in a single chip, muti-core virtualiuzation has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers. Application programs must be parallelized to use all cores fully, and software must explicitly assign tasks to the cores, which is a very complex problem. Concerning the first challenge, new programming models, languages, and libraries are needed to make parallel programming easier.

The second challenge has spawned research involving scheduling algorithms and resource management policies

Virtual Hierarchy

A virtual hierarchy is a cache hierarchy that can adapt to fit the workload or mix of workloads. The hierarchy's first level locates data blocks close to the cores needing them for faster access, establishes a shared-cache domain, and establishes a point of coherence for faster communication. The first level can also provide isolation between independent workloads. A miss at the L1 cache can invoke the L2 access.

The following figure illustrates a logical view of such a virtual cluster hierarchy in two



(b) Multiple virtual clusters assigned to various workloads

levels. Each VM operates in a isolated fashion at the first level which minimize both miss access time and performance interference with other workloads or VMs. The second level maintains a globally shared memory facilitates dynamically repartitioning resources without costly cache flushes.

VIRTUAL CLUSTERS AND RESOURCE MANAGEMENT

A physical cluster is a collection of servers interconnected by a physical network such as a LAN whereas virtual clusters have VMs that are interconnected logically by a virtual network across several physical networks. We will study three critical design issues of virtual clusters: live migration of VMs, memory and file migrations, and dynamic deployment of virtual clusters. VMs in virtual cluster have the following interesting properties:

- The virtual cluster nodes can be either physical or virtual machines
- The purpose of using VMs is to consolidate multiple functionalities on the same server
- 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
- 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.

Virtual cluster based on application partitioning or customization. The most important thing is to determine how to store those images in the system efficiently. There are common installations for most users or applications, such as operating systems or user-level programming libraries. These software packages can be preinstalled as templates (called template VMs). With these templates, users can build their own software stacks. New OS instances can be copied from the template VM.

Fast Deployment and Effective Scheduling

Deployment means two things: to construct and distribute software stacks (OS, libraries, applications) to a physical node inside clusters as fast as possible, and to quickly switch runtime environments from one user's virtual cluster to another user's virtual cluster. If one user finishes using his system, the corresponding virtual cluster should shut down or suspend quickly to save the resources to run other VMs for other users

High-Performance Virtual Storage

Basically, there are four steps to deploy a group of VMs onto a target cluster: preparing the disk image, configuring the VMs, choosing the destination nodes, and executing the VM deployment command on every host. Many systems use templates to simplify the disk image preparation process. A template is a disk image that includes a preinstalled operating system with or without certain application software. Templates could implement the COW (Copy on rite) format. A new COW backup file is very small and easy to create and transfer.

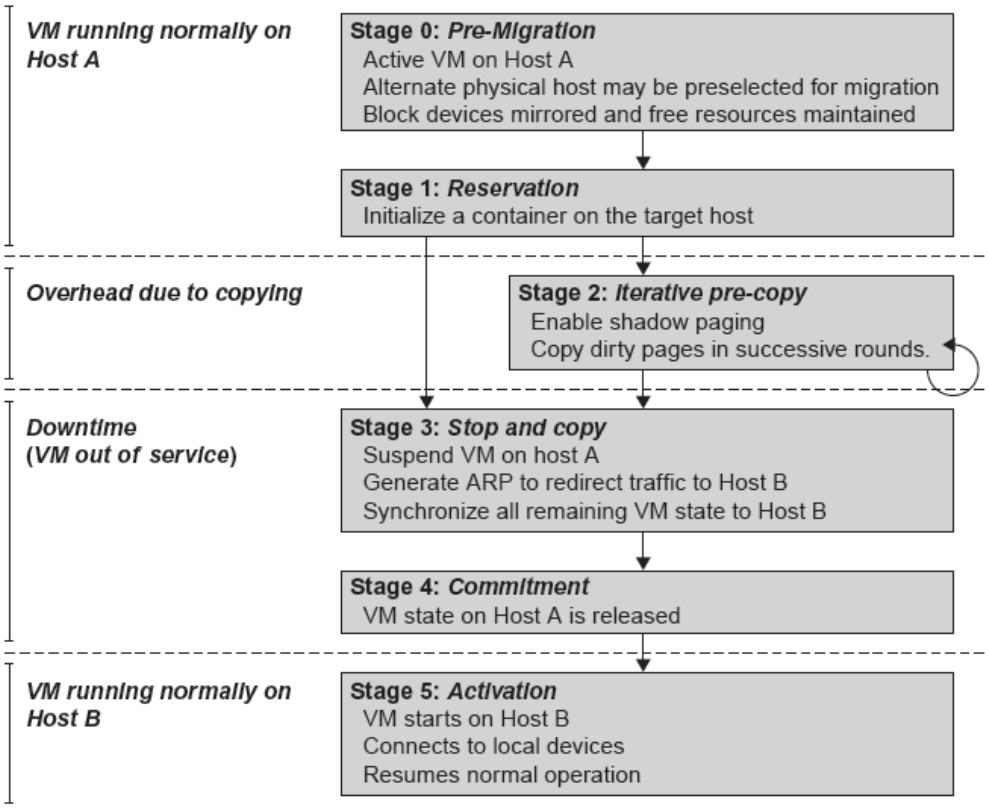
Live VM Migration Steps

There are four ways to manage a virtual cluster. First, we can use a guest-based manager, by which the cluster manager resides on a guest system. In this case, multiple VMs form a virtual cluster. We can build a cluster manager on the host systems. The host-based manager supervises the guest systems and can restart the guest system on another physical machine. Third way to manage a virtual cluster is to use an independent cluster manager on both the host and guest systems. Finally, you can use an integrated cluster on the guest and host systems. This means the manager must be designed to distinguish between virtualized resources and physical resources.

A 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.
- A VM enters the suspended state if its machine file and virtual resources are stored back to the disk.



live migration of a VM from one machine to another 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*
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*
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.*
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*.

Migration of Memory, Files, and Network Resources

When one system migrates to another physical node, we should consider the following issues: Memory migration can be in a range of hundreds of megabytes to a few gigabytes in a typical system today, and it needs to be done in an efficient manner. The Internet Suspend-Resume (ISR) technique exploits temporal locality as memory states are likely to have

considerable overlap in the suspended and the resumed instances of a VM. To exploit temporal locality, each file in the file system is represented as a tree of small subfiles. A copy of this tree exists in both the suspended and resumed VM instances.

File System Migration

Location-independent view of the file system that is available on all hosts. A simple way to achieve this is to provide each VM with its own virtual disk which the file system is mapped to and transport the contents of this virtual disk along with the other states of the VM A distributed file system is used in ISR serving as a transport mechanism for propagating a suspended VM state. The actual file systems themselves are not mapped onto the distributed file system.

Network Migration

To enable remote systems to locate and communicate with a VM, each VM must be assigned a virtual IP address known to other entities. This address can be distinct from the IP address of the host machine where the VM is currently located. Each VM can also have its own distinct virtual MAC address. The VMM maintains a mapping of the virtual IP and MAC addresses to their corresponding VMs. Live migration is a key feature of system virtualization technologies. Here, we focus on VM migration within a cluster environment where a network-accessible storage system, such as storage area network (SAN) or network attached storage (NAS), is employed. Only memory and CPU status needs to be transferred from the source node to the target node. In fact, these issues with the precopy approach are caused by the large amount of transferred data during the whole migration process. A checkpointing/recovery and trace/replay approach (CR/ TR-Motion) is proposed to provide fast VM migration. Another strategy of postcopy is introduced for live migration of VMs. Here, all memory pages are transferred only once during the whole migration process and the baseline total migration time is reduced.

VIRTUALIZATION FOR DATA-CENTER AUTOMATION

Data-center automation refers huge volumes of hardware, software, and database resources in these data centers can be allocated dynamically to millions of Internet users simultaneously, with guaranteed QoS. The latest virtualization development highlights high availability (HA), backup services, workload balancing, and further increases in client bases.

Server Consolidation in Data Centers

The heterogeneous workloads in the data center can be roughly divided into two categories: chatty workloads and noninteractive workloads. Chatty workloads may burst at some point and return to a silent state at some other point. For example, video services can be used by a lot of people at night and few people use it during the day. Noninteractive workloads do not require people's efforts to make progress after they are submitted. Server consolidation is an approach to improve the low utility ratio of hardware resources by reducing the number of physical servers. The use of VMs increases resource management complexity.

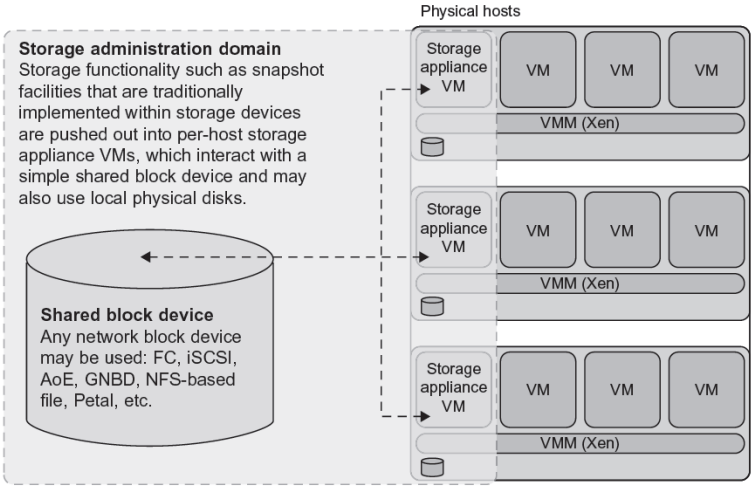
- It enhances hardware utilization. Many underutilized servers are consolidated into fewer servers to enhance resource utilization. Consolidation also facilitates backup services and disaster recovery.
- In a virtual environment, the images of the guest OSes and their applications are readily cloned and reused.
- Total cost of ownership is reduced
- Improves availability and business continuity

Automation of data-center operations includes resource scheduling, architectural support, power management, automatic or autonomic resource management, performance of analytical models, and so on. In virtualized data centers, an efficient, on-demand, fine-grained scheduler is one of the key factors to improve resource utilization. Dynamic CPU allocation is based on VM utilization and application-level QoS metrics. One method considers both CPU and memory flowing as well as automatically adjusting resource overhead based on varying workloads in hosted services. Another scheme uses a two-level resource management system to handle the complexity involved. A local controller at the VM level and a global controller at the server level are designed.

Virtual Storage Management

Virtual storage includes the storage managed by VMMs and guest OSes. Generally, the data stored in this environment can be classified into two categories: VM images and application data. The VM images are special to the virtual environment, while application data includes all other data which is the same as the data in traditional OS environments. In data centers, there are often thousands of VMs, which cause the VM images to become flooded. Parallax is a distributed storage system customized for virtualization environments. Content Addressable Storage (CAS) is a solution to reduce the total size of VM images, and therefore supports a large set of VM-based systems in data centers.

Parallax designs a novel architecture in which storage features that have traditionally been implemented directly on high-end storage arrays and switchers are relocated into a federation of storage VMs. These storage VMs share the same physical hosts as the VMs that they serve. Figure provides an overview of the Parallax system architecture. For each physical machine, Parallax customizes a special storage appliance VM. The storage appliance VM acts as a block virtualization layer between individual VMs and the physical storage device. It provides a virtual disk for each VM on the same physical machine.

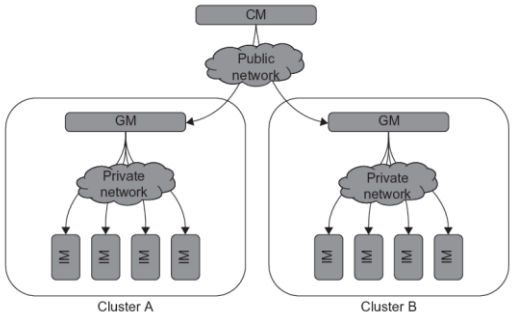


Cloud OS for Virtualized Data Centers

Virtual infrastructure (VI) managers and OSes are specially tailored for virtualizing data centers which often own a large number of servers in clusters. Nimbus, Eucalyptus, and Open Nebula are all open source software available to the public. Only vSphere 4 is a proprietary OS for cloud resource virtualization and management over data centers. These VI managers are used to create VMs and aggregate them into virtual clusters as elastic resources.

Eucalyptus for Virtual Networking of Private Cloud

Eucalyptus is an open source software system (Figure 3.27) intended mainly for supporting Infrastructure as a Service (IaaS) clouds. The system primarily supports virtual networking and the management of VMs



The three resource managers in are specified below:

Instance Manager (IM) controls the execution, inspection, and terminating of VM instances on the host

Group Manager (GM) gathers information about chedules VM execution on specific instance managers, as well as manages virtual instance network.

Cloud Manager (CM) is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes scheduling decisions, and implements them by making requests to group managers.

vSphere 4

vSphere extends earlier virtualization software products by VMware, namely the VMware Workstation, ESX for server virtualization, and Virtual Infrastructure for server clusters overall architecture. The system interacts with user applications via an interface layer, called vCenter. It is primarily intended to offer virtualization support and resource management of data-center resources in building private clouds.

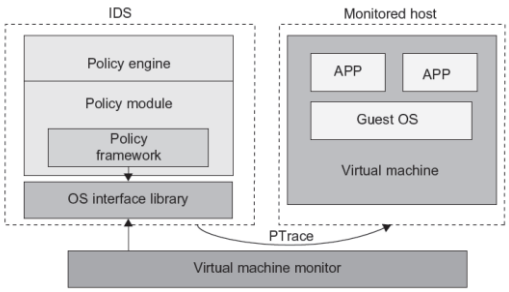
The vSphere 4 is built with two functional software suites: infrastructure services and application services. It also has three component packages intended mainly for virtualization purposes: vCompute is supported by ESX, ESXi, and DRS virtualization libraries from VMware.

Trust Management in Virtualized Data Centers

A VM in the host machine entirely encapsulates the state of the guest operating system running inside it. Encapsulated machine state can be copied and shared over the network and removed like a normal file, which proposes a challenge to VM security. In general, a VMM can provide secure isolation and a VM accesses hardware resources through the control of the VMM, so the VMM is the base of the security of a virtual system. Normally, one VM is taken as a management VM to have some privileges such as creating, suspending, resuming, or deleting a VM.

VM-Based Intrusion Detection

Intrusions are unauthorized access to computer from local/ network users and intrusion detection is used to recognize the unauthorized access. Virtualization-based intrusion detection can isolate guest VMs on the same hardware platform. VMM monitors and audits access requests for hardware and system software. There are two different methods for implementing a VM-based IDS: Either the IDS is an independent process in each VM or a high-privileged VM on the VMM; or the IDS is integrated into the VMM and has the same privilege to access the hardware as well as the VMM. The proposed IDS to run on a VMM as a high-privileged VM is depicted in the following figure.



The VM-based IDS contains a policy engine and a policy module. The policy framework can monitor events in different guest VMs by operating system interface library and PTrace indicates trace to secure policy of monitored host. Besides IDS, honeynets are also prevalent in intrusion detection. They attract and provide a fake system view to attackers in order to protect the real system. In addition, the attack action can be analyzed, and a secure IDS can be built. A honeypot is a purposely defective system that simulates an operating system to cheat and monitor the actions of an attacker.