UNIT -2
VIRTUAL MACHINES AND VIRTUALIZATION OF CLUSTERS AND DATA CENTERS

Implementation Levels of Virtualization, Virtualization Structures/ Tools and mechanisms. Virtualization of CPU, Virtualization of Memory and I/O Devices, Virtual Clusters and Resource Management, Virtualization for Data Center.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The idea of VMs can be dated back to the 1960s. 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. Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers.

LEVELS OF VIRTUALIZATION IMPLEMENTATION*:* A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in below Fig.(a). 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 below Fig.(b). This virtualization layer is known as hypervisor or virtual machine monitor (VMM) . 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.

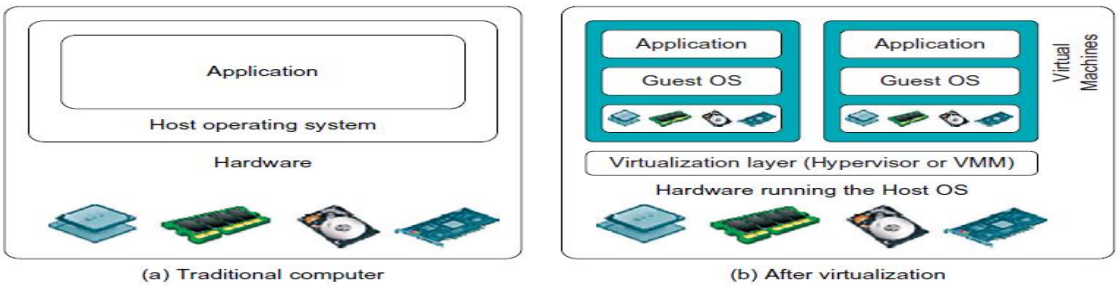


Fig: The architecture of a computer system before and after virtualization

Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level as shown in the below fig.

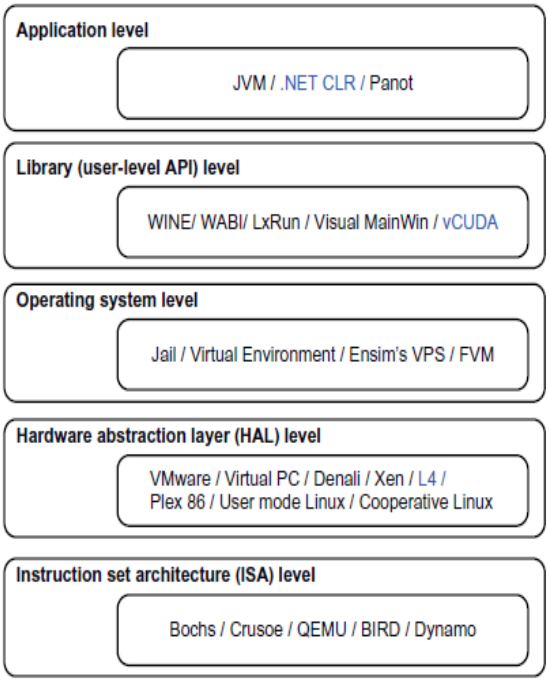


Fig: Virtualization ranging from hardware to applications in five abstraction levels

Instruction Set Architecture Level: At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. 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 required.

Hardware Abstraction Level this kind of virtualization is carried out in order to make the hardware to be available to multiple users. 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.

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 callsby 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.

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,

***VMM DESIGN REQUIREMENTS AND PROVIDERS*:** The below table shows various VMM providers and requirements for the establishments:

Comparison of Four VMM and Hypervisor Software packages				
Providers	**Host CPU**	**Guest OS**	**Host OS**	**Architecture**
VM Ware Server	X86	Same as VM Ware	No host OS	Para Virtualization
VM Ware Workstation	X86	Windows, Solaris, Linux	Linux, Windows	Full Virtualization
KVM	X86	Linux, Windows	Linux	Para Virtualization
Xen	X86	Free BSD	Linux, Solaris	Hypervisor

VIRTUALIZATION SUPPORT AT THE OS LEVEL: With the help of VM technology, a new computing mode known as cloud computing is emerging. Cloud computing is transforming the computing landscape by shifting the hardware and staffing costs of managing a computational center to third parties, just like banks.

Need for OS-Level Virtualization: Due to these drawbacks such as

- As it is slow to initialize a hardware-level VM because each VM creates its own image from scratch. Besides slow operation, storing the VM images also becomes an issue.
- Full virtualization at the hardware level also has the disadvantages of slow performance and low density, and the need for para-virtualization to modify the guest OS.

Operating system 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), Virtual Private System (VPS).

Advantages of OS Extensions : Compared to hardware-level virtualization, the benefits of OS extensions are twofold:

(1) VMs at the operating system level have minimal startup/shutdown costs, low resource requirements, and high scalability

(2) For an OS-level VM, it is possible for a VM and its host environment to synchronize state changes when necessary.

These benefits can be achieved via two mechanisms of OS-level virtualization:

(1) All OS-level VMs on the same physical machine share a single operating system kernel;

(2) The virtualization layer can be designed in a way that allows processes in VMs to access as many resources of the host machine as possible.

Disadvantages of OS Extensions: The main disadvantage of OS extensions is that all the VMs at operating system level on a single container must have the same kind of guest operating system. ***The chroot*** command in a UNIX system can create several virtual root directories within a host OS. These virtual root directories are the root directories of all VMs created. There are two ways to implement virtual root directories: duplicating common resources to each VM partition; or sharing most resources with the host environment and only creating private resource copies on the VM on demand.

Virtualization on Linux or Windows Platforms: OS-level virtualization systems are Linux-based. Virtualization support on the Windows-based platform is still in the research stage. The Linux kernel offers an abstraction layer to allow software processes to work with and operate on resources without knowing the hardware details. New hardware may need a new Linux kernel to support.

MIDDLEWARE SUPPORT FOR VIRTUALIZATION: Library-level virtualization is also known as user-level Application Binary Interface (ABI) or API emulation. This type of virtualization can create execution environments for running alien programs on a platform rather than creating a VM to run the entire operating system. There exist several library-level virtualization systems: namely the Windows Application Binary Interface (WABI), lxrun, WINE, Visual MainWin, and vCUDA, which are summarized in the below Table .

Middleware and library support for virtualization	
WINE	It is a library function which supports the virtualization of x86
Visual MainWin	For developing windows applications it is used as a compiler support
vCUDA	General purpose GPUs are used in order to support for virtualization

WABI	It acts as a middleware which converts windows systems calls running on x86
Lxrun	It is a system call which enables applications written for x86 in linux

VIRTUALIZATION STRUCTURES/ TOOLS AND MECHANISMS

In general, there are three typical classes of VM architecture. Below fig. shows the architectures of a machine before and after virtualization. Before virtualization, the operating system manages the hardware. After virtualization, a virtualization layer is inserted between the hardware and the operating system. In such a case, the virtualization layer is responsible for converting portions of the real hardware into virtual hardware.

HYPERVISOR AND XEN-ARCHITECTURE: The hypervisor supports hardware-level virtualization (see above Fig.(b)) on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor. The hypervisor provides hypercalls for the guest OSes and applications. Depending on the functionality, a hypervisor can assume micro-kernel architecture like the Microsoft Hyper-V.

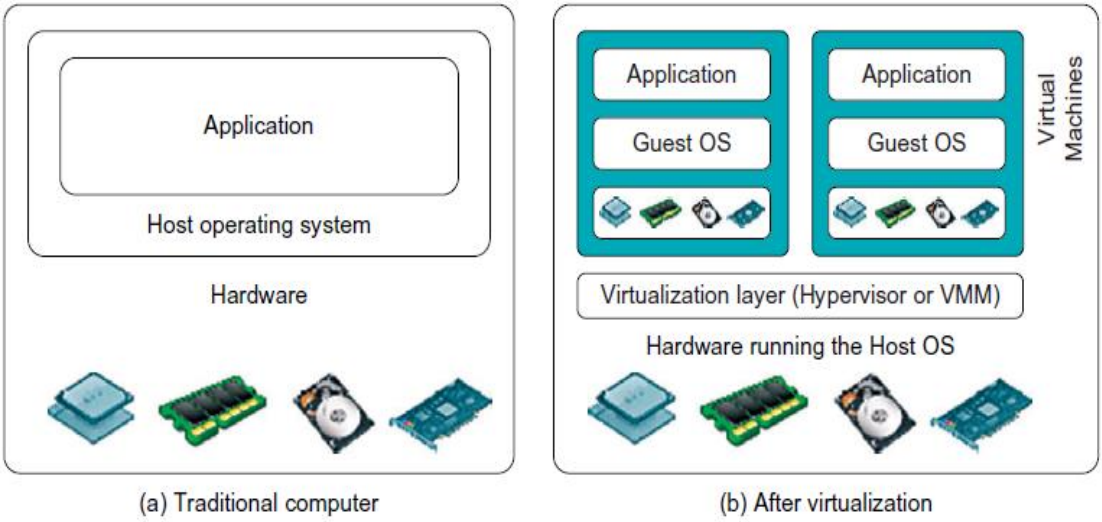


Fig: The architecture of a computer system before and after virtualization

The 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. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0, as shown in below Fig.. 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. As a result, the size of the Xen hypervisor is kept rather small. Xen provides a virtual environment located between the hardware and the OS.

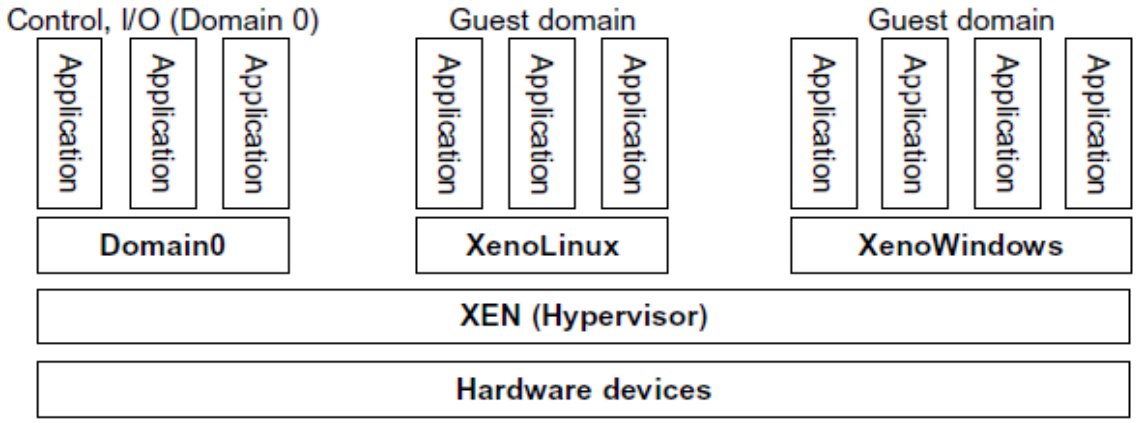
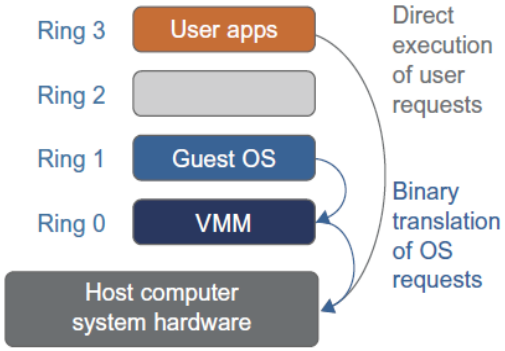


Fig: The Xen architecture's special domain 0 for control and I/O,

The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen. 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. Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains

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 does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, non virtualizable instructions. The guest OSes and their applications consist of noncritical and critical instructions. In a host-based system, both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS.

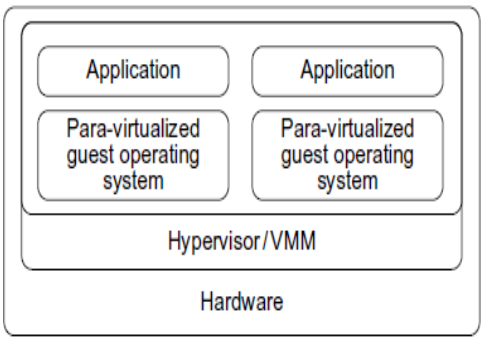
Full Virtualization: With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software. Both the hypervisor and VMM approaches are considered full virtualization.



Binary Translation of Guest OS Requests Using a VMM: This approach was implemented by VMware and many other software companies. As shown in above fig., VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation.

***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. 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. Compared to the hypervisor/VMM architecture, the performance of the host-based architecture may also be low. When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly.

PARA-VIRTUALIZATION WITH COMPILER SUPPORT: Para-virtualization 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. The virtualization layer can be inserted at different positions in a machine software stack. However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.



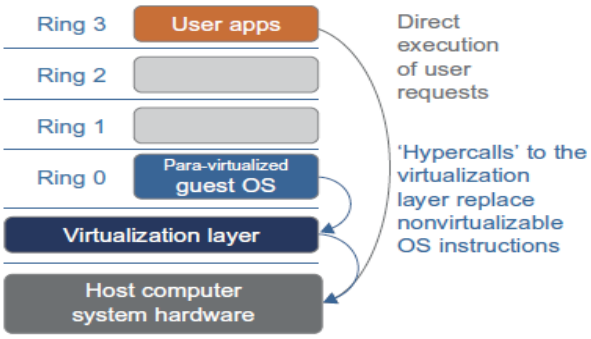


Fig: Para-virtualized VM architecture, The use of a para-virtualized guest OS assisted by an intelligent compiler

Above fig. illustrates the concept of a para-virtualized VM architecture. The guest operating systems are para-virtualized. They are assisted by an intelligent compiler to replace the non virtualizable OS instructions by hyper calls as illustrated in above fig. 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.

Para-Virtualization Architecture: When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS. According to the x 86 ring definitions, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause so e problems. the above fig. show that Para-Virtualization replaces non virtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM. There are certain problems associated with para virtualization as First, its compatibility and portability may be in doubt, because it must support the unmodified OS as well. Second, the cost of maintaining para-virtualized OS's is high, because they may require deep OS kernel modifications.

KVM (Kernel-Based VM): KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants This is a Linux para-virtualization system—a part of the Linux version 2.6.20 kernel. Memory management and scheduling activities are carried out by the existing Linux kernel.

The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine.

***Para-Virtualization with Compiler Support*:** Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handles these instructions at compile time. The guest OS kernel is modified to replace the privileged and sensitive instructions with hyper calls to the hypervisor or VMM. Xen assumes such para-virtualization architecture. The guest OS running in a guest domain may run at Ring 1 instead of at Ring 0. This implies that the guest OS may not be able to execute some privileged and sensitive instructions. The privileged instructions are implemented by hyper-calls to the hypervisor.

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.

***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 instruction.

CPU Virtualization: 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. In this case, the VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system. However, not all CPU architectures are virtualizable. 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.

Memory Virtualization: Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory. All modern x86 CPUs include a memory management unit (MMU) and a translation look aside buffer (TLB) to optimize virtual memory performance.

The VMM is responsible for mapping the guest physical memory to the actual machine memory. as shown in the below fig. shows the two-level memory mapping procedure.

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. Nested page tables add another layer of indirection to virtual memory. 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. Since modern operating systems maintain a set of page tables for every process, the shadow page tables will get flooded.

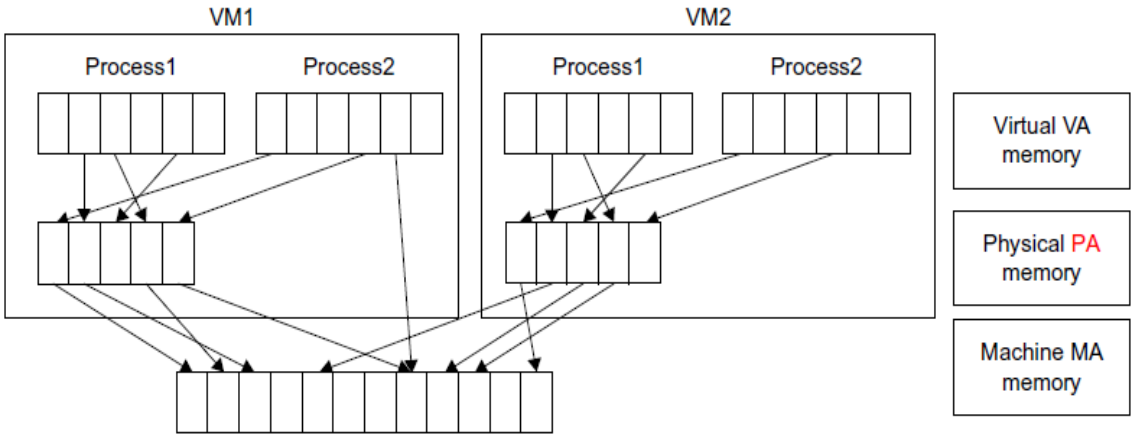


Fig: Two-level memory mapping procedure

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.

I/O Virtualization: I/O virtualization 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, and direct I/O. Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well-known, real-world devices.

Single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates. The para-virtualization method of I/O virtualization is typically used in Xen. Direct I/O virtualization 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.

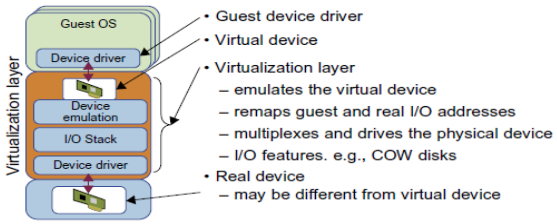
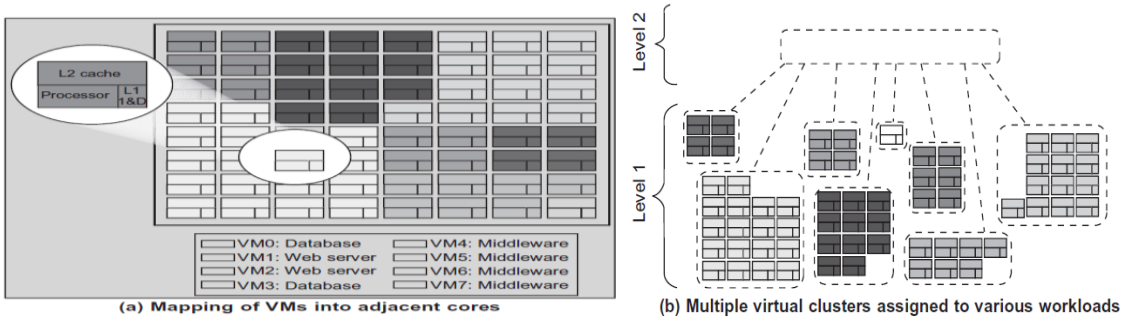


Fig: Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices

VIRTUALIZATION IN MULTI-CORE PROCESSORS: Virtualization of a multi-core processor is relatively more complicated than virtualizing a uni-core processor. Though multicore processors are having higher performance with the integration of multiple processor cores in a single chip, Multi-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. 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. When a miss leaves a tile, it first attempts to locate the block (or sharers) within the first level. The first level can also provide isolation between independent workloads. A miss at the L1 cache can invoke the L2 access. Space sharing is applied to assign three workloads to three clusters of virtual cores: namely VM0 and VM3 for database workload, VM1 and

VM2 for web server workload, and VM4–VM7 for middleware workload. The basic assumption is that each workload runs in its own VM. However, space sharing applies equally within a single operating system, as shown in below fig.



(a) Mapping of VMs into adjacent cores (b) Multiple virtual clusters assigned to various workloads

The fig illustrates a logical view of such a virtual cluster hierarchy in two levels. Each VM operates in a isolated fashion at the first level. This will minimize both miss access time and performance interference with other workloads or VMs.

VIRTUAL CLUSTERS AND RESOURCE MANAGEMENT

A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN. When a traditional VM is initialized, the administrator needs to manually write configuration information or specify the configuration sources. When more VMs join a network, an inefficient configuration always causes problems with overloading or underutilization. Amazon's Elastic Compute Cloud (EC2) is a good example of a web service that provides elastic computing power in a cloud.

Physical versus Virtual Clusters: 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. 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

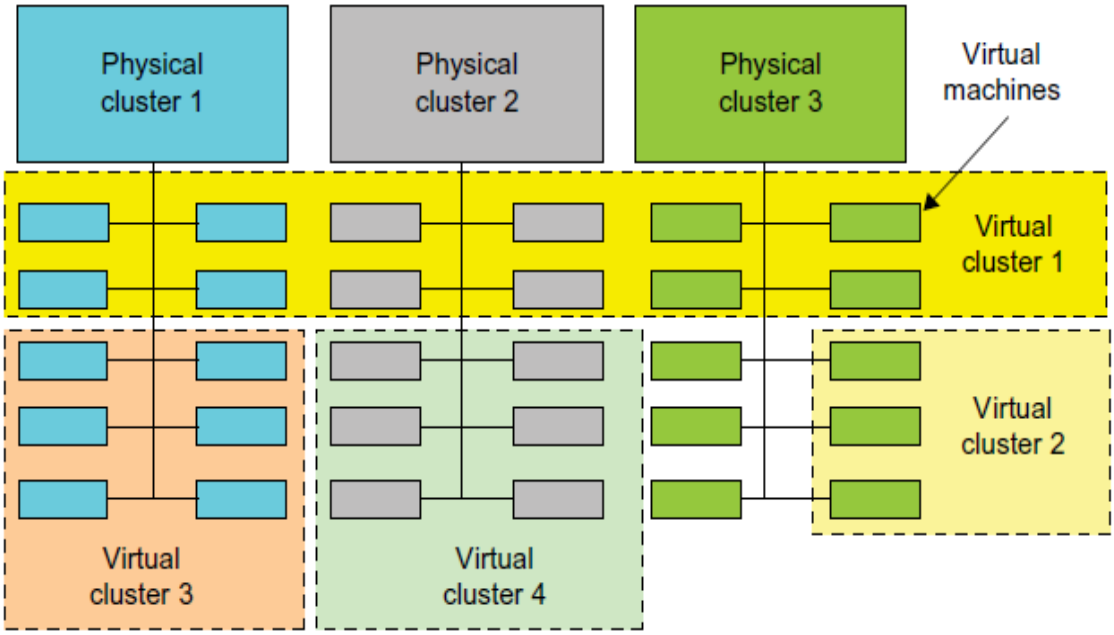


Fig: A cloud platform with four virtual clusters over three physical clusters shaded differently.

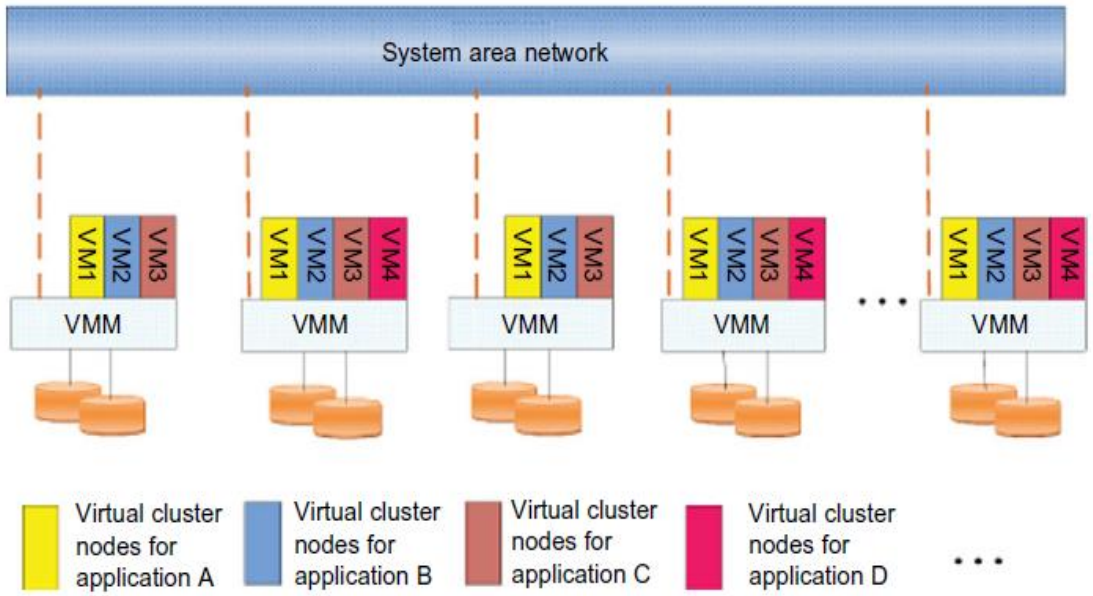


Fig: The concept of a virtual cluster based on application partitioning

Live VM migration Steps and Performance Effects: 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. In other words, a physical node can fail over to a VM on another host. There are four ways to manage a virtual cluster.

- First, one 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.
- Second, one 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. This will make infrastructure management more complex, however.
- Finally, one 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.

Migration of Memory , Files and Network Resources : here the following three types of migrations are described which are given as follows:

Memory migration: This is one of the most important aspects of VM migration. Moving the memory instance of a VM from one physical host to another can be

approached in any number of ways. But traditionally, the concepts behind the techniques tend to share common implementation paradigms. The techniques employed for this purpose depend upon the characteristics of application / workloads supported by the guest OS. 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.

File migration: To support VM migration, a system must provide each VM with a consistent, 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. However, due to the current trend of high capacity disks, migration of the contents of an entire disk over a network is not a viable solution. Another way is to have a global file system across all machines where a VM could be located.

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. Instead, the VMM only accesses its local file system. The relevant VM files are explicitly copied into the local file system for a resume operation and taken out of the local file system for a suspend operation.

Network migration: If the source and destination machines of a VM migration are typically connected to a single switched LAN, an unsolicited ARP reply from the migrating host is provided advertising that the IP has moved to a new location. This solves the open network connection problem by reconfiguring all the peers to send future packets to a new location.

Dynamic Deployment of Virtual Clusters: the below tables shows four virtual cluster research projects. Their design objectives and reported results. The Cellular Disco at Stanford is a virtual cluster built in a shared-memory multiprocessor system. The INRIA virtual cluster was built to test parallel algorithm performance

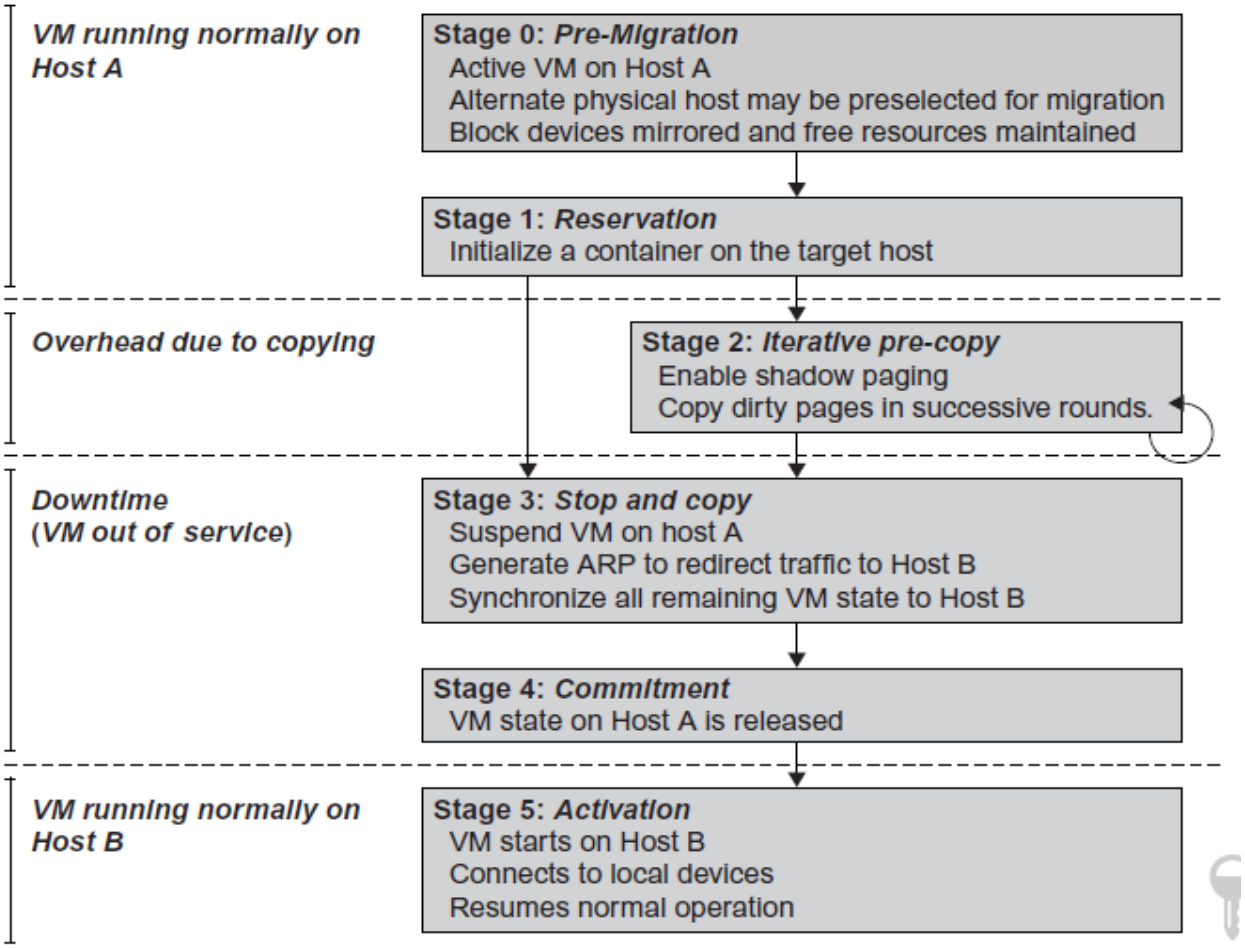


Fig: Live migration process of a VM from one host to another.

Experimental Results on Four Research Virtual Clusters		
Project Name	**Design Objective**	**Reported Results and References**
GRAAL Project at INRIA in France	For Observation of Performance of parallel algorithms	75% of performance
Cluster on Demand at Duke Univ.	Allocation of dynamic resource with virtual cluster management	Multiple virtual clusters sharing VMs
VIOLIN at Purdue Univ.	Advantage Dynamic adaptation	Reduction in execution time
Cellular Disco at Stanford Univ.	Deploying of virtual clusters	Multiple processors deployed on VMs

VIRTUALIZATION FOR DATA CENTER AUTOMATION

Data-center automation means that 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 and cost- effectiveness. The latest virtualization development highlights high availability (HA), backup services, workload balancing, and further increases in client bases. Data centers have grown rapidly in recent years, and all major IT companies are pouring their resources into building new data centers. In addition, Google, Yahoo!, Amazon, Microsoft, HP, Apple, and IBM are all in the game. All these companies have invested billions of dollars in datacenter construction and automation.

This automation process is triggered by the growth of virtualization products and cloud computing services. From 2006 to 2011, according to an IDC 2007 report on the growth of virtualization and its market distribution in major IT sectors. In 2006, virtualization has a market share of $1,044 million in business and enterprise opportunities.

Server Consolidation in Data Centers: In data centers, a large number of heterogeneous workloads can run on servers at various times. These heterogeneous workloads can be roughly divided into two categories: **chatty workloads** and **non interactive** workloads. Chatty workloads may burst at some point and return to a silent state at some other point. A web video service is an example of this, whereby a lot of people use it at night and few people use it during the day. Non interactive workloads do not require people's efforts to make progress after they are submitted. High-performance computing is a typical example of this.

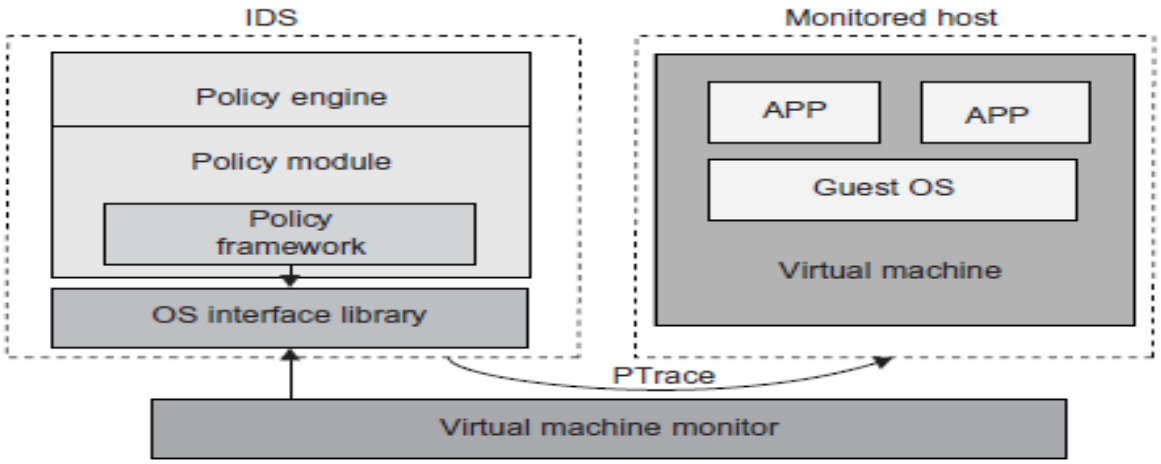


Fig: The architecture of livewire for intrusion detection using a dedicated VM.

Server virtualization has the following side effects:

• Consolidation enhances hardware utilization. Many underutilized servers are consolidated into fewer servers to enhance resource utilization. Consolidation also facilitates backup services and disaster recovery.

• This approach enables more agile provisioning and deployment of resources. In a virtual environment, the images of the guest OSes and their applications are readily cloned and reused.

• The total cost of ownership is reduced. In this sense, server virtualization causes deferred purchases of new servers, a smaller data-center footprint, lower maintenance costs, and lower power, cooling, and cabling requirements.

• This approach improves availability and business continuity. The crash of a guest OS has no effect on the host OS or any other guest OS.

Virtual Storage Management: The term "storage virtualization" was widely used before the renaissance of system virtualization. The most important aspects of system virtualization are encapsulation and isolation. Previously, storage virtualization was largely used to describe the aggregation and repartitioning of disks at very coarse time scales for use by physical machines. In system virtualization, 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 virtualization environments, a virtualization layer is inserted between the hardware and traditional operating systems or a traditional operating system is modified to support virtualization. This procedure complicates storage operations. On the one hand, storage management of the guest OS performs as though it is operating in a real hard disk while the guest OSes cannot access the hard disk directly. On the other hand, many guest OSes contest the hard disk when many VMs are running on a single physical machine.

Cloud OS for Virtualized Data Centers: Data centers must be virtualized to serve as cloud providers. The below table shows a list of OSes These VI managers and OSes are specially tailored for virtualizing data centers which often own a large number of servers in clusters. Nimbus, Eucalyptus, and OpenNebula are all open

source software available to the general 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. Nimbus and Eucalyptus support essentially virtual networks. OpenNebula has additional features to provision dynamic resources and make advance reservations. All three public VI managers apply Xen and KVM for virtualization. vSphere 4 uses the hypervisors ESX and ESXi from VMware.

Managers and Operating systems for Virtualizing Data Centers					
Manager/OS Platform	**Resources Being Virtualized Web Link**	**Hypervisors used**	**Public Cloud Interface**	**Client API**	**Special features**
Eucalyptus	Virtual networking	Xen , KVM	EC2	EC2,WS	Virtual networks
vSphere 4	OS virtualized for data centers	VMWare, ESX	VMWare Vcloud	GUI,WS	Protection of data
Nimbus	Virtual Cluster creation	Xen , KVM	EC2	EC2,WS	Virtual networks
Open Nebula	VM management	Xen , KVM	EC, Elastic Host	XML-RPC	Dynamic provisioning of virtual networks

The three resource managers in the above fig. are specified below:

• Instance Manager Controls the execution, inspection, and terminating of VM instances on the host where it runs.

• Group Manager gathers information about and schedules VM execution on specific instance managers, as well as manages virtual instance network.

• Cloud Manager 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

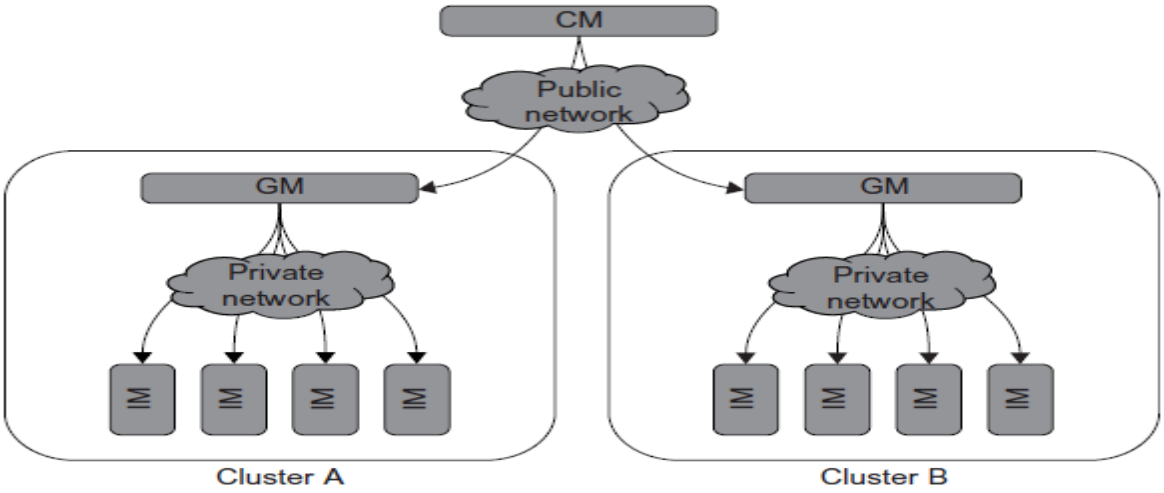


Fig: Eucalyptus for building private clouds by establishing virtual networks over the VMs

TRUST MANAGEMENT IN VIRTUALIZED DATA CENTERS: 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. A VMM changes the computer architecture. It provides a layer of software between the operating systems and system hardware to create one or more VMs on a single physical platform. A VM 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.

Once a hacker successfully enters the VMM or management VM, the whole system is in danger. A subtler problem arises in protocols that rely on the "freshness" of their random number source for generating session keys. Considering a VM, rolling back to a point after a random number has been chosen, but before it has been used, resumes execution; the random number, which must be "fresh" for security purposes, is reused.

VM-Based Intrusion Detection: Intrusions are unauthorized access to a certain computer from local or network users and intrusion detection is used to recognize the unauthorized access. An intrusion detection system (IDS) is built on operating systems, and is based on the characteristics of intrusion actions. A typical IDS can be classified as a ***host-based IDS (HIDS)*** or a ***network-based IDS (NIDS***), depending on the data source. A HIDS can be implemented on the monitored system. When the monitored system is attacked by hackers, the HIDS also faces the risk of

being attacked. A NIDS is based on the flow of network traffic which can't detect fake actions. Virtualization-based intrusion detection can isolate guest VMs on the same hardware platform. Even some VMs can be invaded successfully; they never influence other VMs, which is similar to the way in which a NIDS operates. 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, honeypots and 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. A honeypot can be divided into physical and virtual forms.

UNIT -2
VIRTUAL MACHINES AND VIRTUALIZATION OF CLUSTERS AND DATA CENTERS

Implementation Levels of Virtualization, Virtualization Structures/ Tools and mechanisms. Virtualization of CPU, Virtualization of Memory and I/O Devices, Virtual Clusters and Resource Management, Virtualization for Data Center.

IMPLEMENTATION LEVELS OF VIRTUALIZATION

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The idea of VMs can be dated back to the 1960s. 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. Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers.

LEVELS OF VIRTUALIZATION IMPLEMENTATION*:* A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in below Fig.(a). 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 below Fig.(b). This virtualization layer is known as

hypervisor or virtual machine monitor (VMM) . 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.

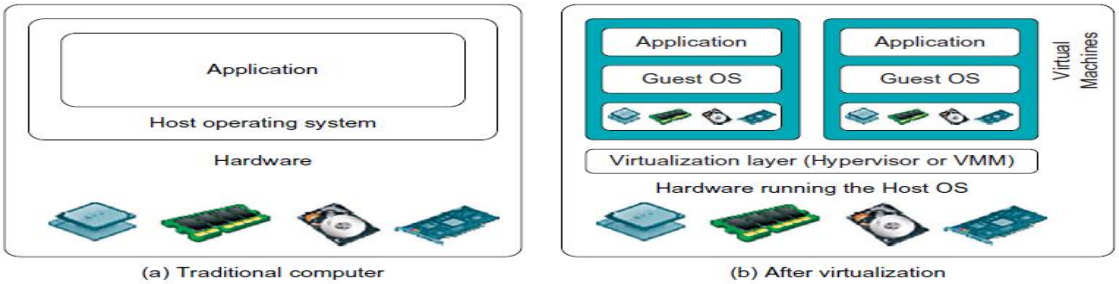
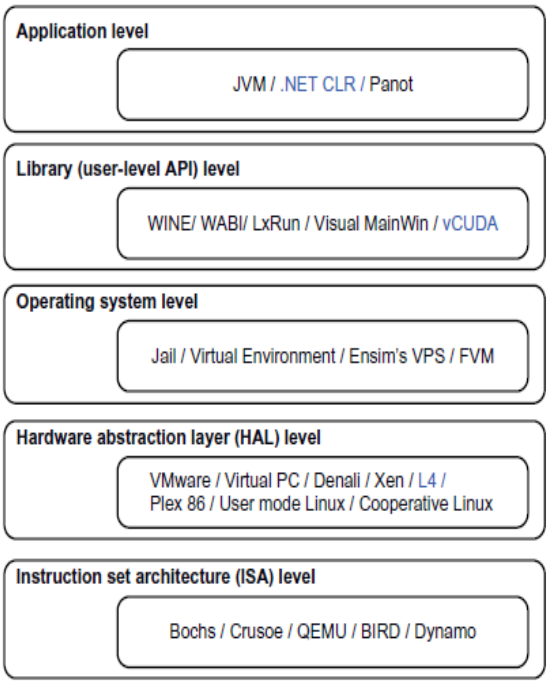


Fig: The architecture of a computer system before and after virtualization

Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level as shown in the below fig.



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 required.

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

Hardware Abstraction Level this kind of virtualization is carried out in order to make the hardware to be available to multiple users. 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.

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 callsby 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.

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,

***VMM DESIGN REQUIREMENTS AND PROVIDERS*:** The below table shows various VMM providers and requirements for the establishments:

Comparison of Four VMM and Hypervisor Software packages				
Providers	**Host CPU**	**Guest OS**	**Host OS**	**Architecture**
VM Ware Server	X86	Same as VM Ware	No host OS	Para Virtualization
VM Ware Workstation	X86	Windows, Solaris, Linux	Linux, Windows	Full Virtualization
KVM	X86	Linux, Windows	Linux	Para Virtualization
Xen	X86	Free BSD	Linux, Solaris	Hypervisor

VIRTUALIZATION SUPPORT AT THE OS LEVEL: With the help of VM technology, a new computing mode known as cloud computing is emerging. Cloud computing is transforming the computing landscape by shifting the hardware and staffing costs of managing a computational center to third parties, just like banks.

Need for OS-Level Virtualization: Due to these drawbacks such as

- As it is slow to initialize a hardware-level VM because each VM creates its own image from scratch. Besides slow operation, storing the VM images also becomes an issue.
- Full virtualization at the hardware level also has the disadvantages of slow performance and low density, and the need for para-virtualization to modify the guest OS.

Operating system 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), Virtual Private System (VPS).

Advantages of OS Extensions : Compared to hardware-level virtualization, the benefits of OS extensions are twofold:

(1) VMs at the operating system level have minimal startup/shutdown costs, low resource requirements, and high scalability

(2) For an OS-level VM, it is possible for a VM and its host environment to synchronize state changes when necessary.

These benefits can be achieved via two mechanisms of OS-level virtualization:

(1) All OS-level VMs on the same physical machine share a single operating system kernel;

(2) The virtualization layer can be designed in a way that allows processes in VMs to access as many resources of the host machine as possible.

Disadvantages of OS Extensions: The main disadvantage of OS extensions is that all the VMs at operating system level on a single container must have the same kind of guest operating system. ***The chroot*** command in a UNIX system can create several virtual root directories within a host OS. These virtual root directories are the root directories of all VMs created. There are two ways to implement virtual root directories: duplicating common resources to each VM partition; or sharing most resources with the host environment and only creating private resource copies on the VM on demand.

Virtualization on Linux or Windows Platforms: OS-level virtualization systems are Linux-based. Virtualization support on the Windows-based platform is still in the research stage. The Linux kernel offers an abstraction layer to allow software

processes to work with and operate on resources without knowing the hardware details. New hardware may need a new Linux kernel to support.

MIDDLEWARE SUPPORT FOR VIRTUALIZATION: Library-level virtualization is also known as user-level Application Binary Interface (ABI) or API emulation. This type of virtualization can create execution environments for running alien programs on a platform rather than creating a VM to run the entire operating system. There exist several library-level virtualization systems: namely the Windows Application Binary Interface (WABI), lxrun, WINE, Visual MainWin, and vCUDA, which are summarized in the below Table .

Middleware and library support for virtualization	
WINE	It is a library function which supports the virtualization of x86
Visual MainWin	For developing windows applications it is used as a compiler support
vCUDA	General purpose GPUs are used in order to support for virtualization
WABI	It acts as a middleware which converts windows systems calls running on x86
Lxrun	It is a system call which enables applications written for x86 in linux

VIRTUALIZATION STRUCTURES/ TOOLS AND MECHANISMS

In general, there are three typical classes of VM architecture. Below fig. shows the architectures of a machine before and after virtualization. Before virtualization, the operating system manages the hardware. After virtualization, a virtualization layer is inserted between the hardware and the operating system. In such a case, the virtualization layer is responsible for converting portions of the real hardware into virtual hardware.

HYPERVISOR AND XEN-ARCHITECTURE: The hypervisor supports hardware-level virtualization (see above Fig.(b)) on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor. The hypervisor provides hypercalls for the guest OSes and applications. Depending on the functionality, a hypervisor can assume micro-kernel architecture like the Microsoft Hyper-V.

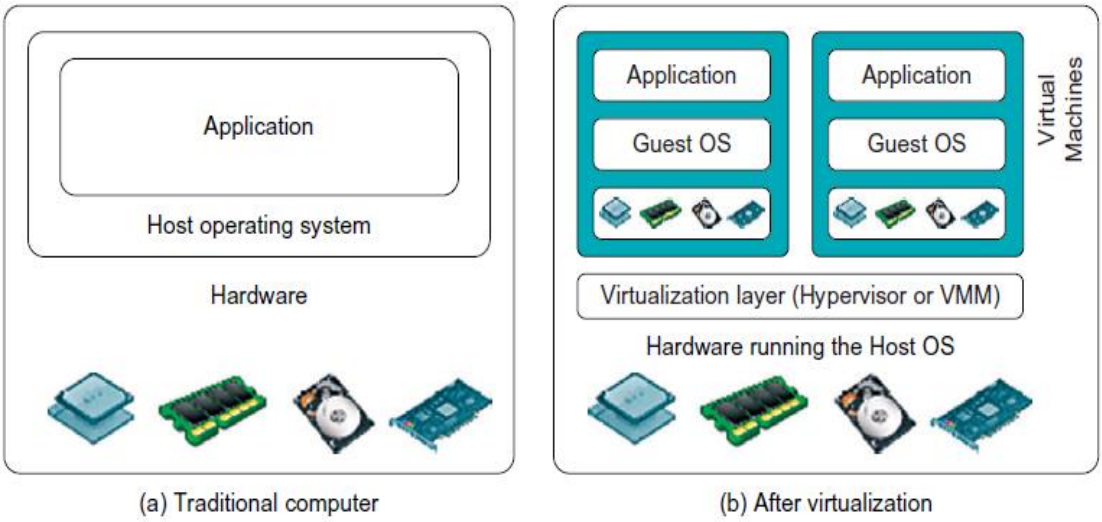


Fig: The architecture of a computer system before and after virtualization

The 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. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0, as shown in below Fig.. 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. As a result, the size of the Xen hypervisor is kept rather small. Xen provides a virtual environment located between the hardware and the OS.

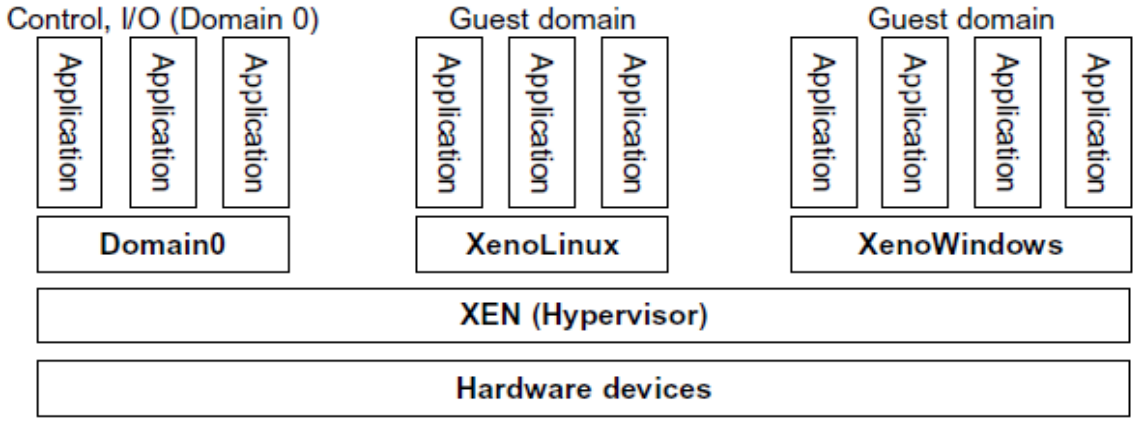


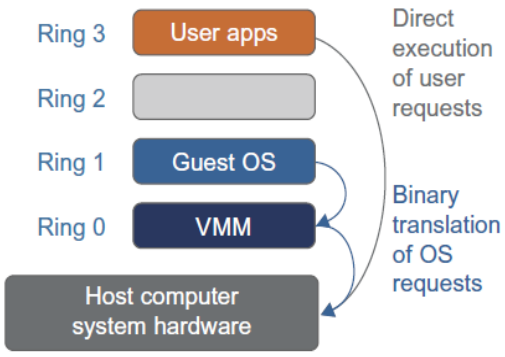
Fig: The Xen architecture's special domain 0 for control and I/O,

The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen. 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. Therefore, one of the

responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains

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 does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, non virtualizable instructions. The guest OSes and their applications consist of noncritical and critical instructions. In a host-based system, both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS.

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.



Binary Translation of Guest OS Requests Using a VMM: This approach was implemented by VMware and many other software companies. As shown in above fig., VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation.

***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. 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. Compared to the hypervisor/VMM architecture, the performance of

the host-based architecture may also be low. When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly.

PARA-VIRTUALIZATION WITH COMPILER SUPPORT: Para-virtualization 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. The virtualization layer can be inserted at different positions in a machine software stack. However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.

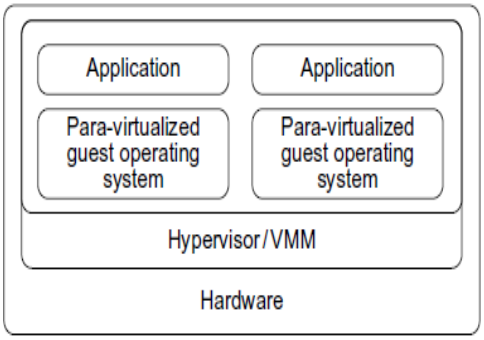
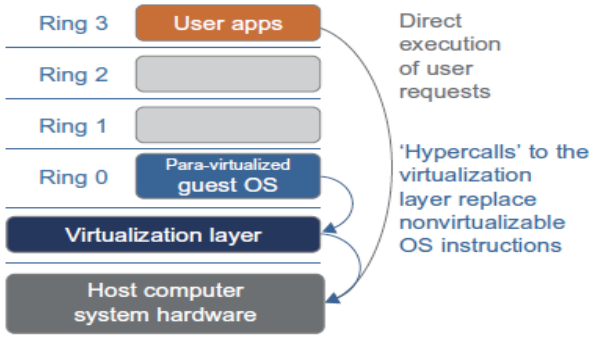


Fig: Para-virtualized VM architecture,



The use of a para-virtualized guest OS assisted by an intelligent compiler

Above fig. illustrates the concept of a para-virtualized VM architecture. The guest operating systems are para-virtualized. They are assisted by an intelligent compiler to replace the non virtualizable OS instructions by hyper calls as illustrated in above fig. 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.

Para-Virtualization Architecture: When the x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS. According to the x 86 ring definitions, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause so e problems. the above fig. show that Para-Virtualization replaces non virtualizable instructions with hypercalls that communicate directly with the hypervisor or VMM. There are certain problems

associated with para virtualization as First, its compatibility and portability may be in doubt, because it must support the unmodified OS as well. Second, the cost of maintaining para-virtualized OS's is high, because they may require deep OS kernel modifications.

KVM (Kernel-Based VM): KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants This is a Linux para-virtualization system—a part of the Linux version 2.6.20 kernel. Memory management and scheduling activities are carried out by the existing Linux kernel. The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine.

***Para-Virtualization with Compiler Support*:** Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handles these instructions at compile time. The guest OS kernel is modified to replace the privileged and sensitive instructions with hyper calls to the hypervisor or VMM. Xen assumes such para-virtualization architecture. The guest OS running in a guest domain may run at Ring 1 instead of at Ring 0. This implies that the guest OS may not be able to execute some privileged and sensitive instructions. The privileged instructions are implemented by hyper-calls to the hypervisor.

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.

***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 instruction.

CPU Virtualization: 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. In this case, the VMM acts as a unified mediator for hardware access from different VMs to guarantee the correctness and stability of the whole system. However, not all CPU architectures are virtualizable. 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.

Memory Virtualization: Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory. All modern x86 CPUs include a memory management unit (MMU) and a translation look aside buffer (TLB) to optimize virtual memory performance.

The VMM is responsible for mapping the guest physical memory to the actual machine memory. as shown in the below fig. shows the two-level memory mapping procedure.

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. Nested page tables add another layer of indirection to virtual memory. 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. Since modern operating systems maintain a set of page tables for every process, the shadow page tables will get flooded.

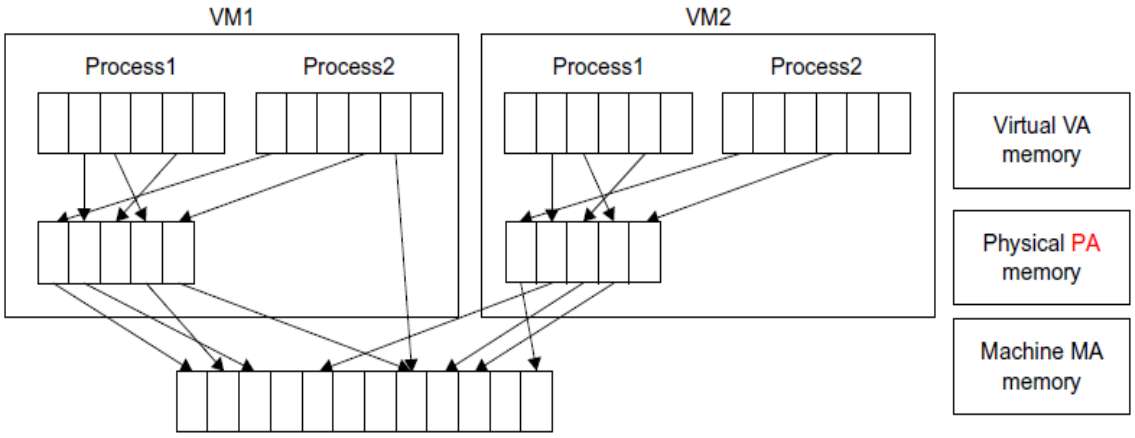


Fig: Two-level memory mapping procedure

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.

I/O Virtualization: I/O virtualization 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, and direct I/O. Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well-known, real-world devices.

Single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates. The para-virtualization method of I/O virtualization is typically used in Xen. Direct I/O virtualization 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.

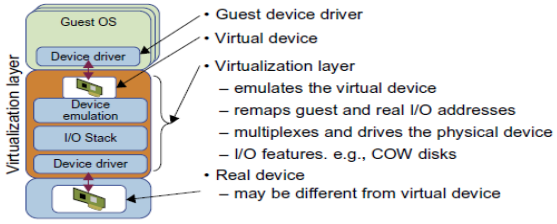
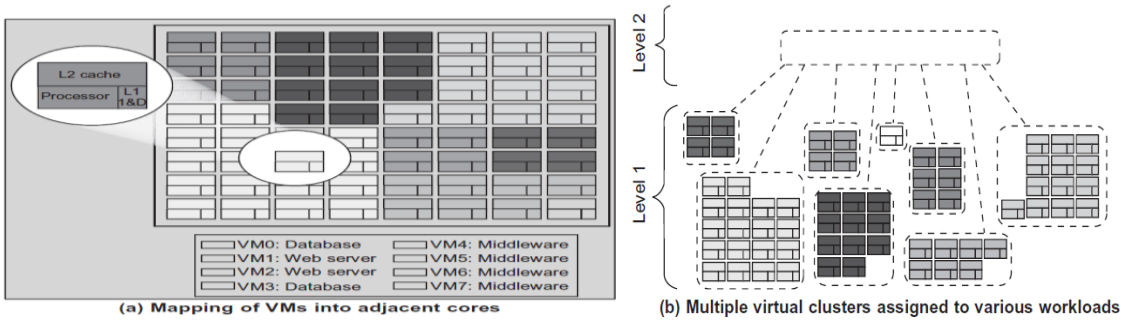


Fig: Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices

VIRTUALIZATION IN MULTI-CORE PROCESSORS: Virtualization of a multi-core processor is relatively more complicated than virtualizing a uni-core

processor. Though multicore processors are having higher performance with the integration of multiple processor cores in a single chip, Multi-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. 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. When a miss leaves a tile, it first attempts to locate the block (or sharers) within the first level. The first level can also provide isolation between independent workloads. A miss at the L1 cache can invoke the L2 access. Space sharing is applied to assign three workloads to three clusters of virtual cores: namely VM0 and VM3 for database workload, VM1 and VM2 for web server workload, and VM4–VM7 for middleware workload. The basic assumption is that each workload runs in its own VM. However, space sharing applies equally within a single operating system, as shown in below fig.



(a) Mapping of VMs into adjacent cores

(b) Multiple virtual clusters assigned to various workloads

The fig illustrates a logical view of such a virtual cluster hierarchy in two levels. Each VM operates in a isolated fashion at the first level. This will minimize both miss access time and performance interference with other workloads or VMs.

VIRTUAL CLUSTERS AND RESOURCE MANAGEMENT

A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN. When a traditional VM is initialized, the administrator needs to manually write configuration information or specify the configuration sources. When more VMs join a network, an inefficient configuration

always causes problems with overloading or underutilization. Amazon's Elastic Compute Cloud (EC2) is a good example of a web service that provides elastic computing power in a cloud.

Physical versus Virtual Clusters: 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. 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

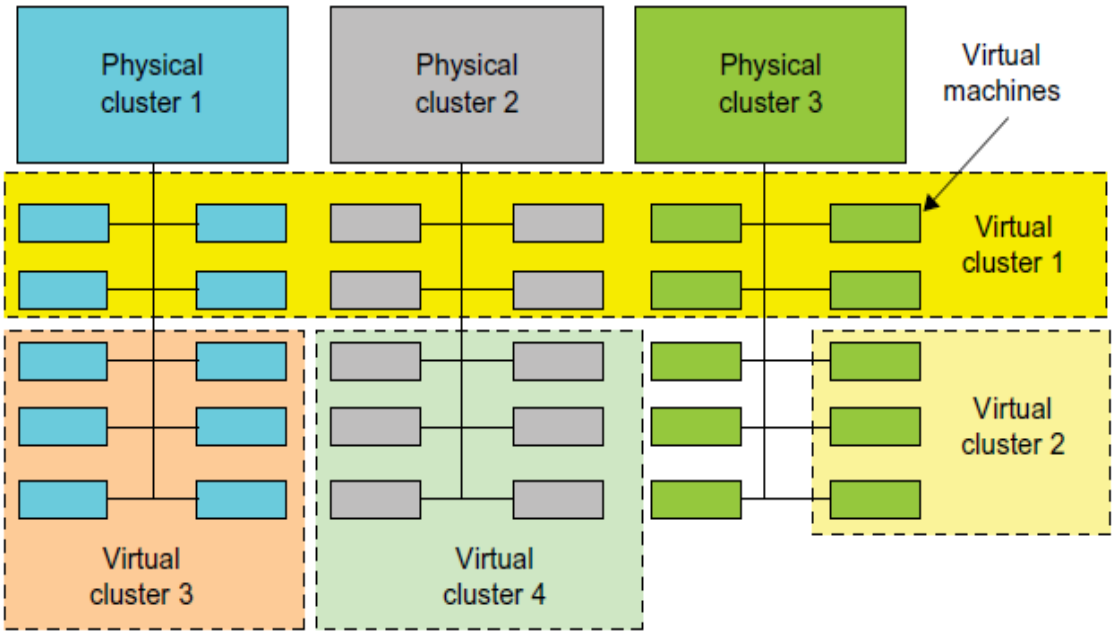


Fig: A cloud platform with four virtual clusters over three physical clusters shaded differently.

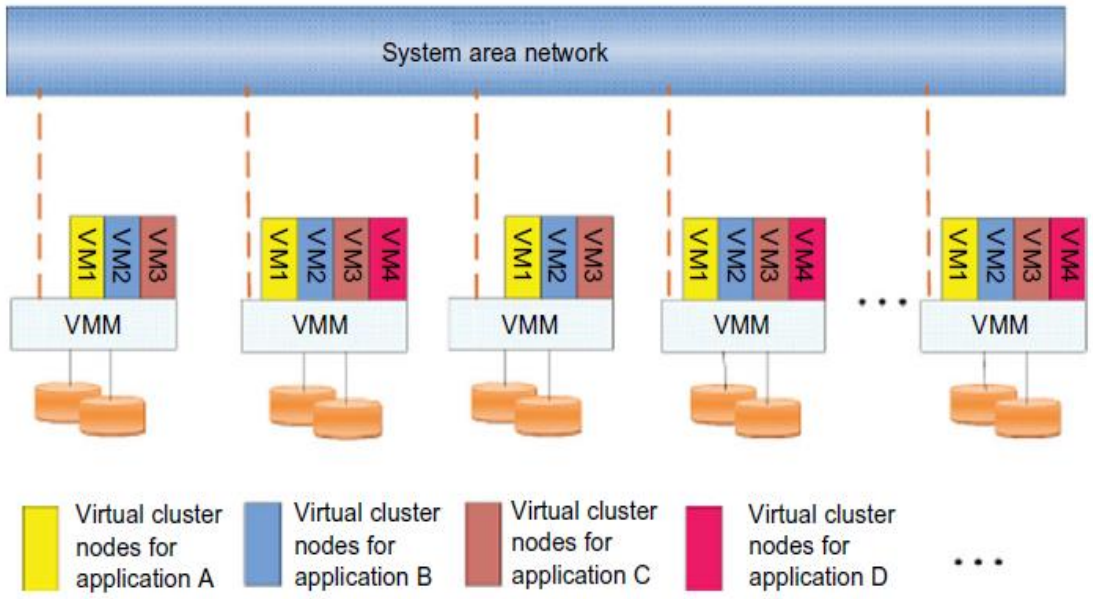


Fig: The concept of a virtual cluster based on application partitioning

Live VM migration Steps and Performance Effects: 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. In other words, a physical node can fail over to a VM on another host. There are four ways to manage a virtual cluster.

- First, one 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.
- Second, one 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. This will make infrastructure management more complex, however.
- Finally, one 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.

Migration of Memory , Files and Network Resources : here the following three types of migrations are described which are given as follows:

Memory migration: This is one of the most important aspects of VM migration. Moving the memory instance of a VM from one physical host to another can be approached in any number of ways. But traditionally, the concepts behind the techniques tend to share common implementation paradigms. The techniques employed for this purpose depend upon the characteristics of application / workloads supported by the guest OS. 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.

File migration: To support VM migration, a system must provide each VM with a consistent, 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. However, due to the current trend of high capacity disks, migration of the contents of an entire disk over a network is not a viable solution. Another way is to have a global file system across all machines where a VM could be located.

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. Instead, the VMM only accesses its local

file system. The relevant VM files are explicitly copied into the local file system for a resume operation and taken out of the local file system for a suspend operation.

Network migration: If the source and destination machines of a VM migration are typically connected to a single switched LAN, an unsolicited ARP reply from the migrating host is provided advertising that the IP has moved to a new location. This solves the open network connection problem by reconfiguring all the peers to send future packets to a new location.

Dynamic Deployment of Virtual Clusters: the below tables shows four virtual cluster research projects. Their design objectives and reported results. The Cellular Disco at Stanford is a virtual cluster built in a shared-memory multiprocessor system. The INRIA virtual cluster was built to test parallel algorithm performance

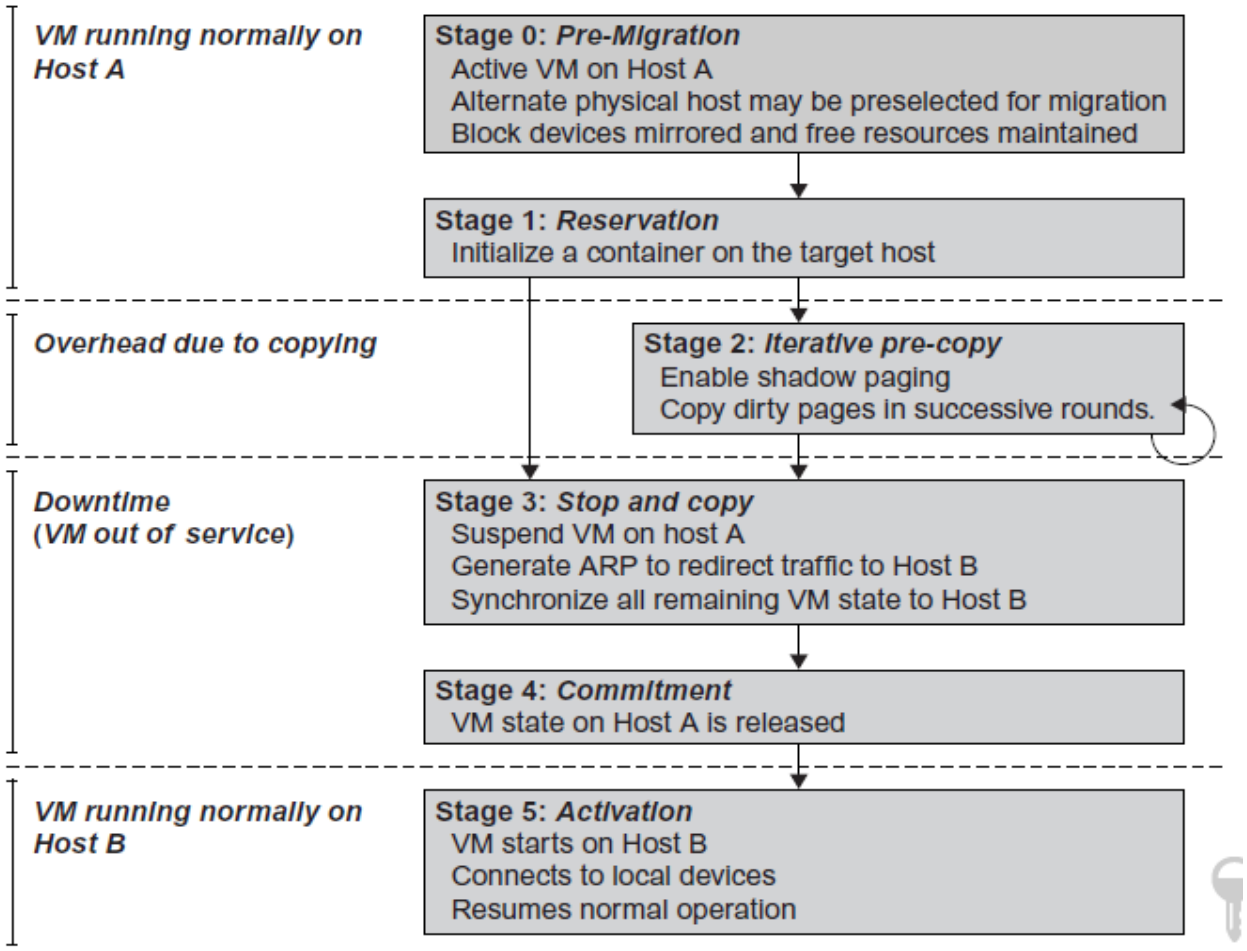


Fig: Live migration process of a VM from one host to another.

Experimental Results on Four Research Virtual Clusters		
Project Name	**Design Objective**	**Reported Results and References**
GRAAL Project at INRIA in France	For Observation of Performance of parallel algorithms	75% of performance
Cluster on Demand at Duke Univ.	Allocation of dynamic resource with virtual cluster management	Multiple virtual clusters sharing VMs
VIOLIN at Purdue Univ.	Advantage Dynamic adaptation	Reduction in execution time
Cellular Disco at Stanford Univ.	Deploying of virtual clusters	Multiple processors deployed on VMs

VIRTUALIZATION FOR DATA CENTER AUTOMATION

Data-center automation means that 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 and cost- effectiveness. The latest virtualization development highlights high availability (HA), backup services, workload balancing, and further increases in client bases. Data centers have grown rapidly in recent years, and all major IT companies are pouring their resources into building new data centers. In addition, Google, Yahoo!, Amazon, Microsoft, HP, Apple, and IBM are all in the game. All these companies have invested billions of dollars in datacenter construction and automation.

This automation process is triggered by the growth of virtualization products and cloud computing services. From 2006 to 2011, according to an IDC 2007 report on the growth of virtualization and its market distribution in major IT sectors. In 2006, virtualization has a market share of $1,044 million in business and enterprise opportunities.

Server Consolidation in Data Centers: In data centers, a large number of heterogeneous workloads can run on servers at various times. These heterogeneous workloads can be roughly divided into two categories: **chatty workloads** and **non interactive** workloads. Chatty workloads may burst at some point and return to a silent state at some other point. A web video service is an example of this, whereby a lot of people use it at night and few people use it during the day. Non interactive

workloads do not require people's efforts to make progress after they are submitted. High-performance computing is a typical example of this.

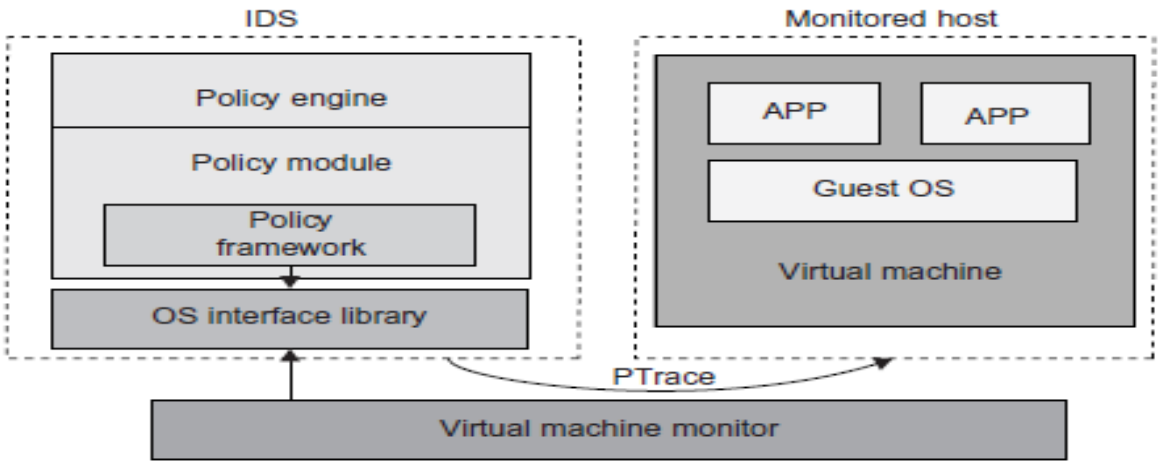


Fig: The architecture of livewire for intrusion detection using a dedicated VM.

Server virtualization has the following side effects:

• Consolidation enhances hardware utilization. Many underutilized servers are consolidated into fewer servers to enhance resource utilization. Consolidation also facilitates backup services and disaster recovery.

• This approach enables more agile provisioning and deployment of resources. In a virtual environment, the images of the guest OSes and their applications are readily cloned and reused.

• The total cost of ownership is reduced. In this sense, server virtualization causes deferred purchases of new servers, a smaller data-center footprint, lower maintenance costs, and lower power, cooling, and cabling requirements.

• This approach improves availability and business continuity. The crash of a guest OS has no effect on the host OS or any other guest OS.

Virtual Storage Management: The term "storage virtualization" was widely used before the renaissance of system virtualization. The most important aspects of system virtualization are encapsulation and isolation. Previously, storage virtualization was largely used to describe the aggregation and repartitioning of disks at very coarse time scales for use by physical machines. In system virtualization, 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 virtualization environments, a virtualization layer is inserted between the hardware and traditional operating systems or a traditional operating system is modified to support virtualization. This procedure complicates storage operations. On the one hand, storage management of the guest OS performs as though it is operating in a real hard disk while the guest OSes cannot access the hard disk directly. On the other hand, many guest OSes contest the hard disk when many VMs are running on a single physical machine.

Cloud OS for Virtualized Data Centers: Data centers must be virtualized to serve as cloud providers. The below table shows a list of OSes These VI managers and OSes are specially tailored for virtualizing data centers which often own a large number of servers in clusters. Nimbus, Eucalyptus, and OpenNebula are all open source software available to the general 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. Nimbus and Eucalyptus support essentially virtual networks. OpenNebula has additional features to provision dynamic resources and make advance reservations. All three public VI managers apply Xen and KVM for virtualization. vSphere 4 uses the hypervisors ESX and ESXi from VMware.

Managers and Operating systems for Virtualizing Data Centers					
Manager/OS Platform	**Resources Being Virtualized Web Link**	**Hypervisors used**	**Public Cloud Interface**	**Client API**	**Special features**
Eucalyptus	Virtual networking	Xen , KVM	EC2	EC2,WS	Virtual networks
vSphere 4	OS virtualized for data centers	VMWare, ESX	VMWare Vcloud	GUI,WS	Protection of data
Nimbus	Virtual Cluster creation	Xen , KVM	EC2	EC2,WS	Virtual networks
Open Nebula	VM management	Xen , KVM	EC, Elastic Host	XML-RPC	Dynamic provisioning of virtual networks

The three resource managers in the above fig. are specified below:

• Instance Manager Controls the execution, inspection, and terminating of VM instances on the host where it runs.

• Group Manager gathers information about and schedules VM execution on specific instance managers, as well as manages virtual instance network.

• Cloud Manager 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

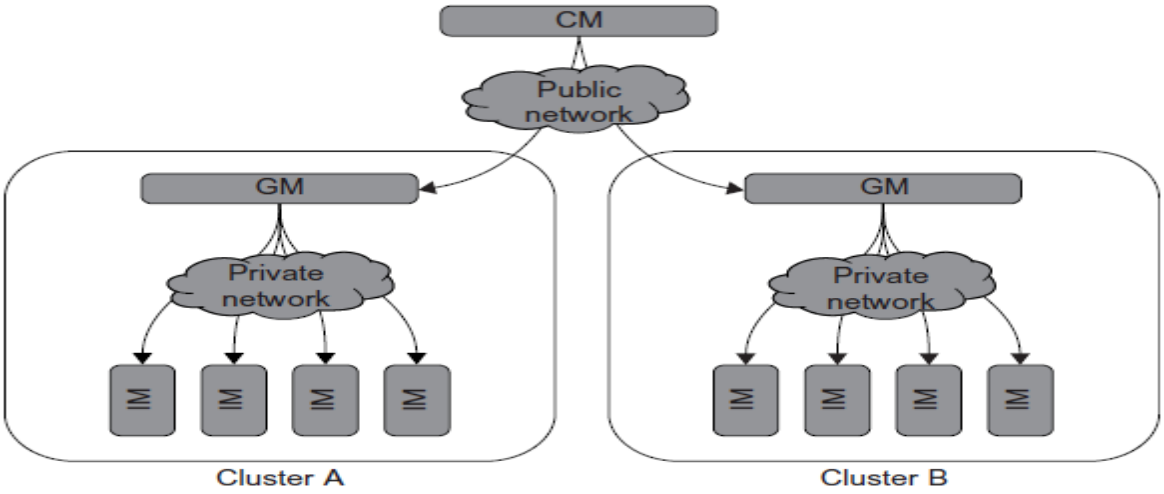


Fig: Eucalyptus for building private clouds by establishing virtual networks over the VMs

TRUST MANAGEMENT IN VIRTUALIZED DATA CENTERS: 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. A VMM changes the computer architecture. It provides a layer of software between the operating systems and system hardware to create one or more VMs on a single physical platform. A VM 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.

Once a hacker successfully enters the VMM or management VM, the whole system is in danger. A subtler problem arises in protocols that rely on the "freshness" of their random number source for generating session keys. Considering a VM, rolling back to a point after a random number has been chosen, but before it has been used, resumes execution; the random number, which must be "fresh" for security purposes, is reused.

VM-Based Intrusion Detection: Intrusions are unauthorized access to a certain computer from local or network users and intrusion detection is used to recognize the unauthorized access. An intrusion detection system (IDS) is built on operating

systems, and is based on the characteristics of intrusion actions. A typical IDS can be classified as a ***host-based IDS (HIDS)*** or a ***network-based IDS (NIDS***), depending on the data source. A HIDS can be implemented on the monitored system. When the monitored system is attacked by hackers, the HIDS also faces the risk of being attacked. A NIDS is based on the flow of network traffic which can't detect fake actions. Virtualization-based intrusion detection can isolate guest VMs on the same hardware platform. Even some VMs can be invaded successfully; they never influence other VMs, which is similar to the way in which a NIDS operates. 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, honeypots and 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. A honeypot can be divided into physical and virtual forms.