



Applied Accelerated AI
Dr. Satyajit Das
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Lecture - 05
Introduction to Operating Systems, Virtualization, Cloud Part -2

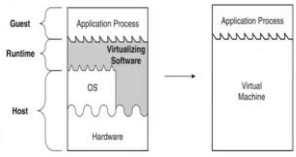
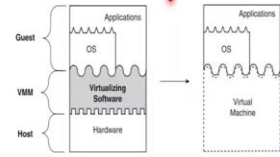
(Refer Slide Time: 00:17)



Different types of VMs

- **Process VM**
 - Virtualizes the ABI
 - Virtualization software => Runtime
 - Runs in non-privileged mode (user space)
 - Performs binary translation.
 - Terminates when guest process terminates.

- **System VM**
 - Virtualizes the ISA
 - Virtualization software => Hypervisor
 - Runs in privileged mode
 - Traps and emulates privileged instructions

(Refer Slide Time: 00:18)

System VM

Hypervisors

- Also called Virtual Machine Monitor (VMM)
- A hypervisor is an operating system for operating systems
 - Provides a virtual execution environment for an entire OS and its applications
 - Controls access to hardware resources
 - When guest OS executes a privileged instruction, Hypervisor intercepts the instruction, checks for correctness and emulates the instruction.

Apps	Apps	Apps
Guest OS 1	Guest OS 2	Guest OS 3
Hypervisor		
Hardware		

So, system VMs in detail so, that we can get some more idea about these stacks. So, basically the system VMs or system virtual machines are also called as hypervisors or

Virtual Machine Monitor or VMMs. So, wherever you see this or encounter this term as VMM or hypervisor; that means, it is the system of virtual machine that is running underneath.

Now, what is an what is a hypervisor? Hypervisor is an operating system for operating system, because you can see that you have this hardware and when you boot this entire system, the first thing gets booted up is this hypervisor. And, then you have multiple OSs which will share the underlying hardware through this hypervisor of course, but these virtual machines will have different OSs let us see OS 1, 2 and 3.

Now, this hypervisor will get booted up initially or in the beginning and then OS 1 will get booted up, OS 2 will get booted up, OS 3 will get booted up ok. So, hypervisor is essentially providing the virtual execution environment for entire OS and its application which will be running on top of this operating system. Now, what was the advantage or what was the difference between this hypervisor and the OS from the system the computing system point of view?

Because, if you have the operating system alone on the top of hardware then the OS was supposed to get booted up in the beginning itself and instead now the hypervisor is getting booted up. And, after that your guest OSs will get booted up on these different virtual machines.

Now, the key difference is that hypervisor here try will try to get invisible or try to stay completely transparent. Because, if you see the operating system point of view operating system will let us say will try to access the hardware through several privileged instructions, that we have seen before like trying to access the file or trying to access the network or so on and so forth.

So, basically you are initiating the privileged instructions from this OS to for this hardware. Now, to get invisible what this hypervisor would try to do is that it will trap those privileged instructions and then it will emulate in the underlying hardware ok, for this underlying hardware which is the underlying eyes. So, guest operating system does not know at all that it is actually doing something else which it was not supposed to do.

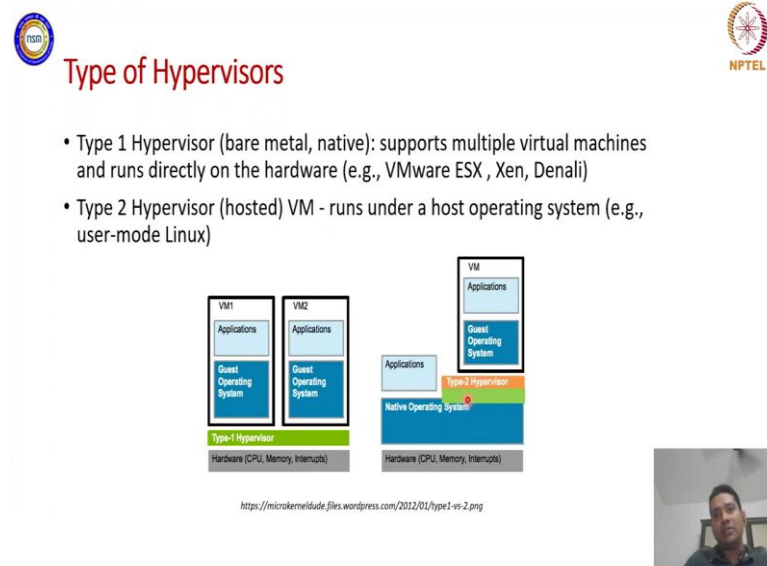
It was supposed to let us say try to access the memory or maybe disk or network interface then it will or maybe let us say some IO devices if you want to access maybe

some maybe let us see your microphone or camera, it wants to try to access then it will initiate the privileged instructions and the hypervisor will trap them and emulate them for the underlying.

And that is the flexibility of using this hypervisor as the completely transparent system between this OS and hardware. So, all the privileged instructions that are coming from different virtual machines hypervisor can now schedule them for the underlying resources that are available. So, the hypervisor is completely doing the same thing as operating system was supposed to do that scheduling, the resource management, the allocation.

So, everything hypervisor is now doing the same things, but for the virtual machines that is on top of that. So, when OS guest OS gets executed that privileged instructions, hypervisors intercepts the instructions and check for correctness and then emulate that instruction for the underlying hardware and that is the difference between a traditional OS and your hypervisor.

(Refer Slide Time: 04:53)



Now, let us look about different types of hypervisors that are available. So, there can be type 1 hypervisor which is on bare metal or native. So, basically hypervisor is lying on top of your bare metal hardware and then it is kind of trapping all the privileged instructions coming from the virtual machines, let us say VM1 and VM2 guest operating systems and applications are running completely in separate environment right.

So, VMware ESX basically VMware has different hypervisors for different machines like desktop; your server. So, basically VMware ESX is server level system hypervisor, Xen also provides system level hypervisor which is type 1 hypervisor. So, basically what these hypervisors doing is that it tries to completely be transparent.

So, the virtual machines or the guest operating system which is running in the virtual machines they will do the they will initiate or invoke these privileged instructions to these hypervisors. Actually, they do not know that they are doing that to hypervisors, but hypervisor underlying that will trap them and emulate them and get back to this one.

So; that means, all the things like basically when you are seeing that the stack it is very important to understand that when you are booting up your hypervisor is get gets booted up initially. So, in the beginning and then your guest operating systems will get so; that means, to have or to control the hardware underlying hardware, the all the IO available inside this hardware, all the network or file system that are available inside the hardware to have the control you need the drivers also present inside the hypervisor.

So, writing hypervisor from scratch is kind of the type 1 hypervisor because it is completely independent of everything and it is underlying all the guest operating system and basically it is sitting on top of your hardware. So, so it must have all the drivers that will be necessary to access the underlying hardware. Now, type 2 hypervisor will give you some flexibility in terms of implementation.

So, what that does is that it is now no longer a part of your; that means, you are now no longer writing the hypervisor from the scratch. Now, what you are doing there, you are now making the hypervisor a part of your operating system as well as part of your user space. So, now, the question is why I will write all the drivers that are already present inside the operating system right.

Operating system has its own drivers and modules to control the underlying hardware. So, why I will write the all the things from the scratch which is the type 1 hypervisor doing. So, with the help of native operating system; so, native operating system is essentially which is running on the bare metal hardware and guest operating system which is running inside your virtual machine which is not running in the on the top of your bare metal hardware.

Now, on the bare metal hardware your native operating system will be running. So, why not that taking the advantage of all the drivers and modules that are present inside the operating system and put your hypervisor in the kernel portion. So, you write one kernel module inside the operating system to take over the privileges; that means, to take over the control of all these modules which will be inside your native operating system.

And, also provide some user level libraries or some links to run your user level applications right. So, now, we have type 2 hypervisor which is kind of hosted on the native operating system. So, that is why it is called also called the hosted virtual machine. Now, you have this native operating system; so, basically all the native applications can be run on this native operating system.

And, if you have the virtual machine that can be on top of your type 2 hypervisor and all these privileged instructions coming from this guest operating system can be trapped and emulated inside this hypervisor and that is the type 2 hypervisor here. Of course, writing this type of hypervisor will not need much of effort because you are not writing from the scratch by implementing all the driver modules for the for controlling the underlying hardware right.

(Refer Slide Time: 10:08)



Para-virtualized VMs



- Modify guest OS for better performance
- Traditional Hypervisors provide full-virtualization
 - They expose to VMs virtual hardware that is functionally identical to the underlying physical hardware.
 - Advantage : allows unmodified guest OS to execute
 - Disadvantage: Sensitive instructions must be trapped and emulated by Hypervisor.
 - E.g. KVM and VMWare ESX provide full virtualization
- Para-virtualized VM
 - Sees a virtual hardware abstraction that is similar, but not identical to the real hardware.
 - Guest OS is modified to replace sensitive instructions with "hypercalls" to the Hypervisor.
 - Advantage: Results in lower performance overhead
 - Disadvantage: Needs modification to the guest OS.
 - Xen provides both para-virtual as well as full-virtualization
- Often traditional Hypervisors are partially para-virtualized
 - Device drivers in guest OS may be para-virtualized whereas CPU and Memory may be fully virtualized.



There are another sets of virtual machines called para-virtualized virtual machines which modifies the. So, basically virtual machine we have seen that modifying the guest OS will give better performance for the para-virtualized virtual machines, because if you see

the traditional hypervisor with full virtualization. So, basically they expose the virtual machines to virtual hardware through this which is fully underlying hardware.

And of course, advantage is that you do not need to modify your guest OS. So, guest OS can run as it is because it does not know that there is underlying hypervisor is running, because the hypervisor which is the full virtualization type 1 hypervisor. So, that is completely running underneath and completely transparent; so, basically you do not need to modify your guest OS ok.

But, disadvantage of this kind of hypervisors is that sensitive instructions must get trapped and emulated by the hypervisor. And, all the sensitive instructions that are coming from your guest OS trapping them and emulating them will cost this resources. So, virtualizations like KVM, the Kernel level Virtual Machine, VMware, ESX that we have seen.

So, all these are providing you full virtualization, but of course, they cannot add up to the entire or all the resources take advantage all the resources that is available, because of this penalty of trapping and emulating right. So, para-virtualized VMs sees a virtual hardware abstraction that is similar to this, but not actually the real hardware.

So, basically what it is doing, guest OS is modified with the replacing some calls; sensitive instruction calls. So, basically you have hardware, the hypervisor and the OS. So, if the hypervisor wants to get fully transparent then all the instructions privileged instructions coming from your OS needs to get trapped into the hypervisor and get simulated.

Now, you are saying that I do not want all the instructions to get emulated inside your hypervisor rather you can have some of these sensitive instructions or define one set of the sensitive instructions which will have executed through this hyper calls to your hypervisor. So, now, hypervisor is no longer transparent.

Because, now OS knows that this set of instructions or this set of sensitive instructions will be through hyper calls to your hypervisor and this set of instructions will be normal system calls right. So, in that way your hypervisor is now no longer fully transparent rather it can capture some set of sensitive instructions through these hyper calls.

So, there will be now interface of this hyper call interface that we have seen that will come into picture now for these para-virtualized virtual machines. And, providers like Xen provides both para-virtual and as well as full virtual machines for the users ok. And, also this was another advantage because often this traditional hypervisor that we have seen are partially para-virtualized.

Because, the device drivers in guest OS may be para-virtualized to the hypervisor calls right. So, some mix and match you can do and you can have this para-virtualized VM. So, this is basically completely orthogonal to your type 1 and type 2 hypervisor that we have seen in the previous slide ok.

(Refer Slide Time: 14:45)

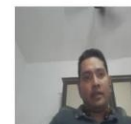


Examples of hypervisors



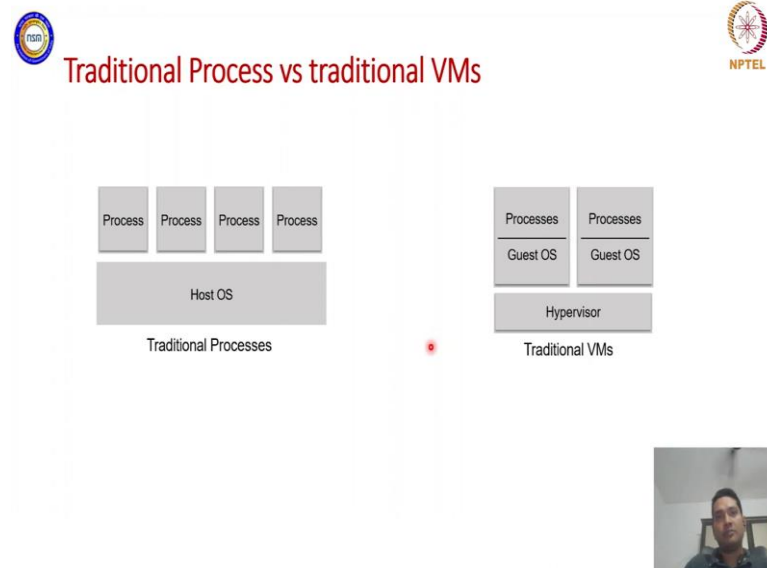
Name	Host ISA	Guest ISA	Host OS	guest OS	Company
Integrity VM	x86-64	x86-64	HP-Unix	Linux, Windows HP Unix	HP
Power VM	Power	Power	No host OS	Linux, AIX	IBM
z/VM	z-ISA	z-ISA	No host OS	Linux on z-ISA	IBM
Linux Secure	x86	x86	No host OS	Linux, Windows	LinuxWorks
Hyper-V Server	x86-64	x86-64	Windows	Windows	Microsoft
Oracle VM	x86, x86-64	x86, x86-64	No host OS	Linux, Windows	Oracle
RTS Hypervisor	x86	x86	No host OS	Linux, Windows	Real Time Systems
SUN zVM	x86, SPARC	same as host	No host OS	Linux, Windows	SUN
VMware ESX Server	x86, x86-64	x86, x86-64	No host OS	Linux, Windows Solaris, FreeBSD	VMware
VMware Fusion	x86, x86-64	x86, x86-64	MAC OS x86	Linux, Windows Solaris, FreeBSD	VMware
VMware Server	x86, x86-64	x86, x86-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Workstation	x86, x86-64	x86, x86-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Player	x86, x86-64	x86, x86-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
Denali	x86	x86	Denali	ILVACO, NetBSD	University of Washington
Xen	x86, x86-64	x86, x86-64	Linux, Solaris	Linux, Solaris NetBSD	University of Cambridge

Credit: Anil Madhavapeddy



So, now next we will talk about we will see some of these hypervisors that are available. So, these are providers VMware, Denali, Xen, IBM, Oracle, Power VM and you can see what kind of host ISA can support and guest ISA they can support, host OS they can support and guest OS they can support and these are the companies by which these names have being provided ok.

(Refer Slide Time: 15:14)



So, now we have seen the two extremes of virtualization. So, basically you have a traditional processes. So, let us say you have hardware, host OS and similar processes are running on top of that. Now, these processes are sharing the underlying resources and complete multiplexing is happening inside these OS to share the resource between these processes. They may be taking one or several threads and all these things will be actually controlled by the OS to have access to this privileged instruction for this hardware access underlying there.

Now, what kind of isolation we have? Of course, the processes can see that all the resources that is available for me, but of course, some virtualization or multiplexing that is done by this OS in this level of multiplexing you can see that several process will be there, but we can see that entire resources are forbidden. But, the OS will handle which portion will go to him and which resources will be allocated for which processes.

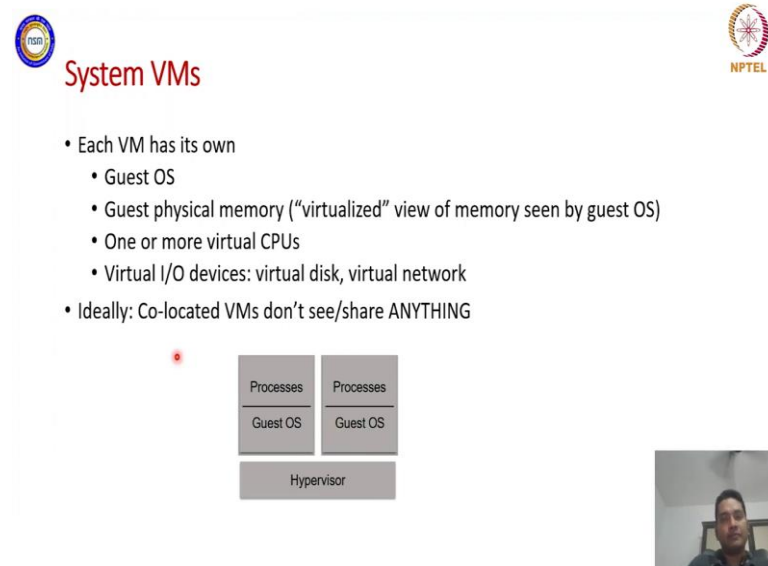
If they are taking too much time they will be preempted and all these things will happen inside the OS. Now, these are very loosely isolated we can see that because let us say these processes will talk to OS through system calls. And, they have let us say 400 near 400 system calls in Linux. Now, you have all the system calls available for all the processes that is available on top of it right so; that means, they are very loosely isolated. So, this is one extreme and in the spectrum another extreme is the traditional virtual machines that we have seen.

So, hypervisor will be there and top of on top of that we will have virtual machines which will have let us say guest OS 1, guest OS 2 and on top of that guest OS you will have similar processes running. Now, these processes running on these guest OS will have no idea about what kind of resources or the isolation between the processes that are running here on top of these two guest OS is completely rigid.

Because, the resources that have been shared between these guest OS are completely managed by this hypervisor right. So, there is a complete isolation of how much disks or resources that will be accessed by this virtual machine on guest OS. So, all this guest OS will see the hardware and this process will only know that this whatever system calls that this process is invoking for this guest OS, no other system calls are there inside this guest OS from this process right.

Because, these processes system calls will be routed through this guest OS which is completely other virtual machine. So, the isolation here is completely rigid and this is kind of co exchange that we can see here right.

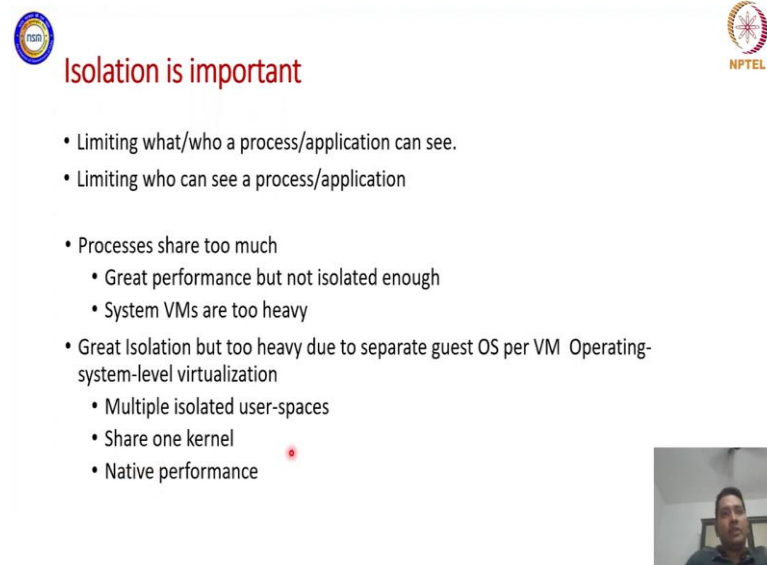
(Refer Slide Time: 18:42)



So, system level virtual machines we can see that each virtual machine has its own guest OS, own guest physical memory which is virtualized view of the memory seen by this guest OS. And, and let us say one or more virtual CPUs based on the allocation by the hypervisor and virtual IO devices which is abstracted by the hypervisor through this virtual machine. Now, ideally so, you can see that these two guest OSs can see or share

anything between them. So, that is the main idea we will be for this system level or system virtual machines.

(Refer Slide Time: 19:26)



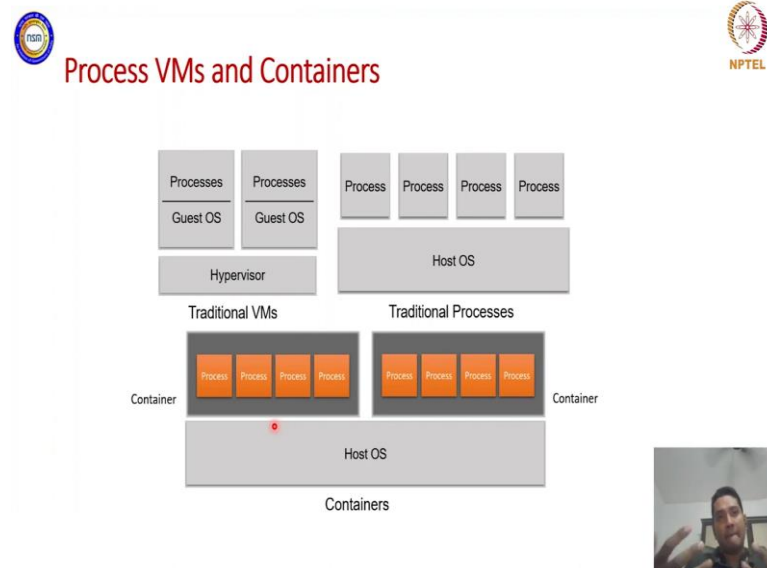
The slide is titled "Isolation is important" in red text. It features a list of bullet points discussing process isolation. In the top right corner, there is a small video inset showing a man speaking. The slide also includes logos for MIT and NPTEL in the top corners.

- Limiting what/who a process/application can see.
- Limiting who can see a process/application
- Processes share too much
 - Great performance but not isolated enough
 - System VMs are too heavy
- Great Isolation but too heavy due to separate guest OS per VM Operating-system-level virtualization
 - Multiple isolated user-spaces
 - Share one kernel
 - Native performance

So, but of course, isolation of these processes are very important because with increase in workload of different application domains what happens is that some processes we need isolated completely from the set of applications. So, limiting of what we want to process, who will want to process, how much resources they can see, how much resources these processes can share.

So, all these are very necessary in terms of or from the point of view of isolation. So, the processes that are sharing of course, they can share too much because their isolation is not high and great isolation will be provided by the system level virtualization through this multiple isolated user spaces and sharing of one kernel and native performance ok.

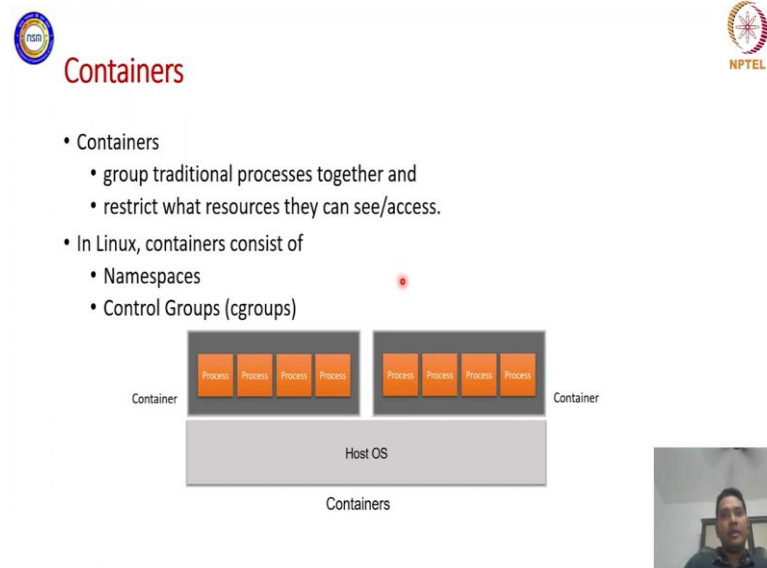
(Refer Slide Time: 20:26)



So, what we will do is that we will go in between of or the midway of these two extremes which is the container based approach. So, this is also one way of virtualizing things, but the thing is that here we are containerizing several processes which we think that these processes can talk to each other.

But, they need to be completely separate from or there should be some isolation between these sets of processes and these sets of processes. And, these are called containers and this container concept was coined in the year of 2008, when Linux operating system started using the namespaces because processes share namespaces.

(Refer Slide Time: 21:14)



So, if you contain the processes in terms of namespaces and control groups then you can have isolated execution of concepts of processes from another sets of processes. So, containers group traditional processes together and restrict what resources they can see and access. So, basically if you are familiarized with the namespaces, you can see that processes share namespaces. So, if you create one separate namespace and control group for one set of process, they can create one container and you can contain this process into these control groups and namespaces.

So, this concept is very important to understand the neat level of virtualization where we cannot have the high or rigid, you can see still container is one process. Because, container will invoke system calls to this host OS and then host OS to your hardware right and also this container will invoke the host system calls to resource OS and so on and so forth.

So, in terms of virtual machines; so, fully virtualized virtual machines or system level virtual machines you cannot get that much of isolation. But, since you have much more resources which are allocated for your full isolation of processes, you can have big resource gain in this approach or container based approach where you can have a kind of namespace separated between these processes and kind of isolated execution or isolated resource sharing between these containers ok.

(Refer Slide Time: 23:20)



Cloud Computing



- Virtualized distributed processing, storage, and software resources and a service.
- Delivering computing as a on-demand, pay-as-you-go service.
- “A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” – NIST definition



So, that is that brings us to cloud computing. So, basically we have talked about virtualization at different levels. Now, if you virtualize everything like your processing if you want to virtualize, if you virtualize your storage, if you virtualize your software resource system as a service; that means, you have seen all the software resources on the entire software stack.

If you virtualize that also ok in terms of packing everything into one resource and you share the resources. Now, as you share the resource of the entire software package as a service right. So, now, we have virtualized concept of processing the storage, the software resources everything as service ok.

And, these services now we can have or the service providers which who will provide these services, they can have pay as you go service. Because, let us say how much distributed processing or how much processing you want, how much storage you want, what are the software resources what you want to access based on that you will be charged from the service providers.

So, that is all about the cloud and according to the NISTs definition cloud is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources. Now, computing resources means of course, the networks, servers, storage, application services everything right that can be rapidly provisioned. So, pay as you go and released with minimal management effort. Because, as you are paying for the

services, you do not want to manage or have more access to you just want to just have access on all the management will be done by the service providers.

So, management effort less or minimal management effort or service provider interaction. So, that is the formal definition of cloud which is provided by NIST and widely accepted. And, as the services that we are talking about if the services are provided as let us see infrastructure service.

(Refer Slide Time: 25:38)

The slide is titled "Service models" and features logos for NIST and NPTEL. It lists the following service models and their characteristics:

- IaaS: Infrastructure as a Service
 - Consumer can provision computing resources within provider's infrastructure upon which they can deploy and run arbitrary software, including OS and applications
- PaaS: Platform as a Service
 - Consumer can create custom applications using programming tools supported by the provider and deploy them onto the provider's cloud infrastructure
- SaaS: Software as a Service
 - Consumer uses provider's applications running on provider's cloud infrastructure

Below the list, three boxes summarize the characteristics of each model:

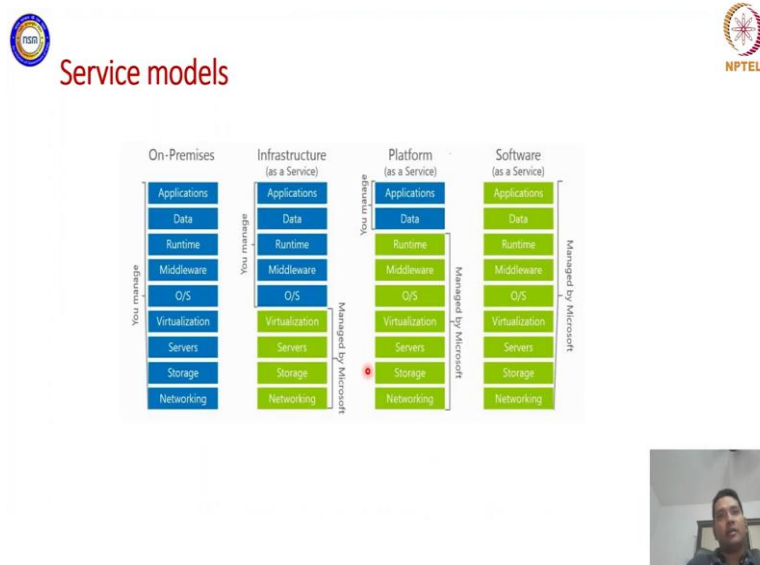
IaaS	PaaS	SaaS
<ul style="list-style-type: none">• Virtual Machines• Virtual Networks	<ul style="list-style-type: none">• Auto Elastic• Continuous Integration	<ul style="list-style-type: none">• Built for Cloud• Uses PaaS

A small video inset in the bottom right corner shows a man speaking.

Basically, customers can then provision the computing resources within the providers infrastructure. So, that will be called as infrastructure as service, you can have platform as service; basically if you or customers can want to create custom applications.

So, have some more flexibility in terms of creating applications to have access to the programming tools supported by the provider. So, this will be called as platform as service and software as service will be the provision where consumer uses providers application running on providers cloud ok. So, this is the end user where you want to try to access the applications they are provided by the service providers.

(Refer Slide Time: 26:32)



So, basically if you see from the different levels of the computing stacks. So, networking, storage, servers, virtualization layers, then operating system, middleware and runtime data and application. So, if you are operating on premise the entire stack of computing system. So, basically you have the entire server stack here and that is managed on premise then you have to manage everything.

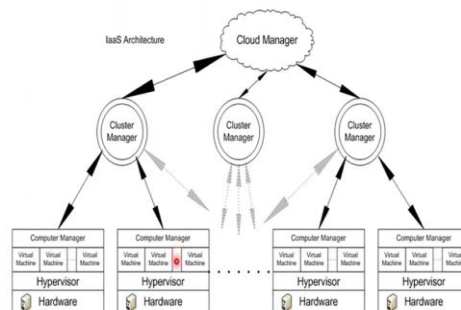
If you are taking infrastructure as a service then you have to manage only the from the software stack; so, after the virtualization layer. So, basically the OS, middleware, runtime, data and application you have to manage and the infrastructure basically which is virtualized servers, storage, networking everything will be managed by the company.

So, this was provided by Microsoft and if you want to have platforms as service only the application and data you want to manage that you also you can have and software as service. So, all the applications only you will access right.

(Refer Slide Time: 27:42)



IAAS cloud architecture



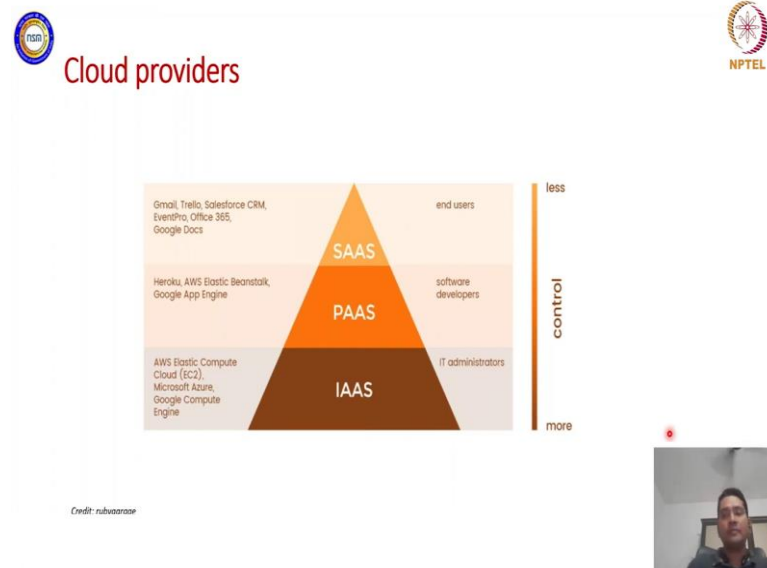
So, just to have an idea how this cloud architecture works from infrastructure point of view, because there you have this the virtual machines which are actually working underneath right.

So, you are the let us say user here and you are talking to the cloud manager, the cloud manager will have a connection between these cluster managers. And, cluster manager will have access to different sets of computing manager which are essentially set of virtual machines which will be running on top of one hypervisor underneath physical hardware that is present.

So, let us say one user is trying to access to some resources in this virtual machine, this virtual machine and another user can access this virtual machine, this virtual machine. But, everybody will talk to your cloud manager through cluster manager and cluster manager will be talking to your computer manager through high bandwidth network.

Now, of course, all the metadata of this accesses, resource allocation and everything will be managed inside your cloud manager who let us say several storage like data center object storage and so on and so forth and the entire service will be managed as such ok. So, as a user you just want to access these machines which are actually provisioned as virtual machines which are managed by this computer manager.

(Refer Slide Time: 29:25)



So, who are the service providers? Just to have a look like to have an idea of what kind of service providers are there for different levels of services. So, infrastructure services are provided by AWS, Amazon, Microsoft, Google computing engine. So, these are mostly accessed by IT administrators, software developers access the platform as service because you want to access the Google app engine or AWS elastic beanstalk or maybe Heroku.

So, basically from the applications point of view users want to access Gmail, let us say CRM service or maybe Office 365, Google docs; all these will be provided as applications as service. So, these are some examples of usage, but we have talked about the operating system, the computing system stack, the virtualization, different types of virtualizations available.

And, you can have actually virtualization in all resource, all levels of resources and you can have the cloud and different cloud providers will provide you different levels of the access.

(Refer Slide Time: 30:37)



The slide features a header with two logos: a circular logo on the left and the NPTEL logo on the right. The title 'Further reading' is in red. Below it, a bullet point lists 'The Architecture of Virtual Machines' as a link. A small video inset in the bottom right corner shows a man speaking.

As for the reading this piece of PDF, you can download from internet and you can read it. It will be very useful to understand part of the virtual machines that are available and how they operate.

So, broad that level of things that we have talked about here will be is actually written here very elaboratively by Smith and Nair. So, this is very useful document, if you want to keep it in your tool chest. So, that is all about it.

Thank you.