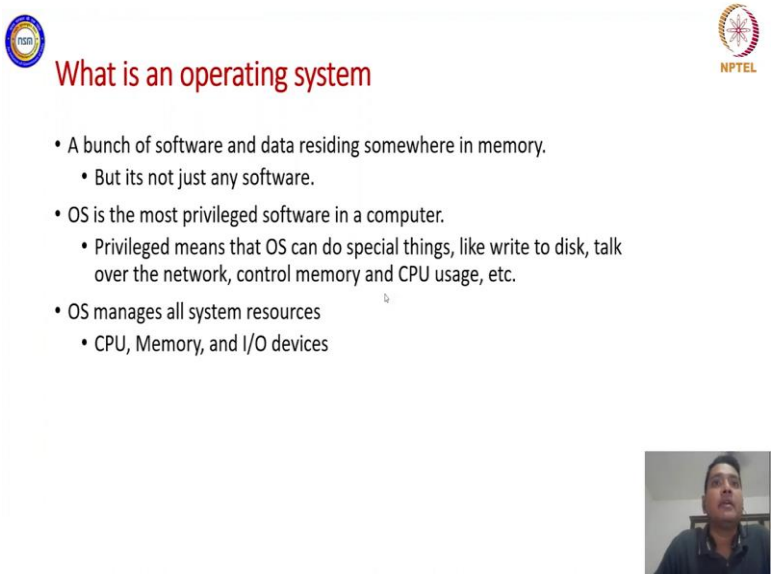
Applied Accelerated AI
Dr. Satyajit Das
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Lecture - 04
Introduction to Operating Systems, Virtualization, Cloud Part - 1

Hello good evening everybody. So, today we will start with the Operating System first. So, we will have a brief overview of what operating system does and what are the privileges that the operating system or the piece of software wants to access. And also we will talk about the virtualization, so this will be the most important aspect of this presentation because we will learn what are the underlying technologies or methodologies that are available for virtualization.

So, that you will get to know when we will use several virtualization methods afterwards you will get to know, what actually you are doing also you will get an overview of the cloud. So, let us start with the operating systems.

(Refer Slide Time: 01:07)

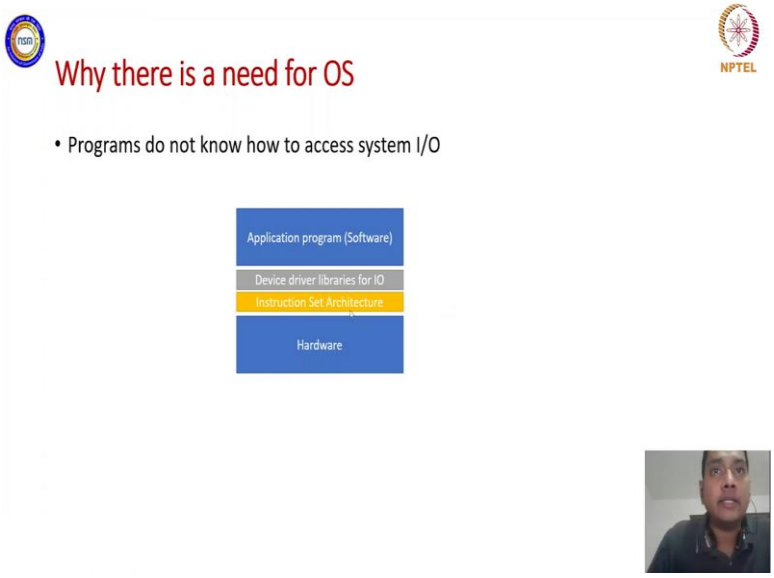


So, operating system is nothing but a piece of software. So, basically it is same as the software that you are having as user applications or user software as you can say. So, but the main difference is that, so they are all together residing in somewhere in memory. But the main but the operating system is not just any other software.

So, operating system or OS is the most privileged software in a computer system. So, privileged means now operating system can do some special things which the user software or user applications cannot do. So, which are these special things like writing to disks, accessing the network cards that is available inside your computing system, control over memory controlling the entire memory hierarchy, CPU usage and so many other things.

Now, as I was mentioning that all the underlying hardware in the computing systems, so you will have had an idea before that of what are the underlying hardware or resources that are available in the computing system of which are like, the processing system like CPU, the storage, the memory and IO devices for peripheral purposes. So, all these underlying hardware will be managed by this operating system this piece of software.

(Refer Slide Time: 02:50)



So, let us have a look why we will need this access or why we will need this software where we can directly run our application software on the hardware. So, now by hardware what I mean? By hardware I mean the resources that are available in a computing system, as the computing system the memory system and other IO devices right.

Now you have your application program which is the piece of software that you want to run. You could run the application program directly onto the software the hardware. So, what you need you need one interface called instruction set architecture. So, this piece of

interface will give you the interface between the software and hardware, so that you can talk to the hardware. That means, controlling the underlying hardware the computing system the memory IO devices.

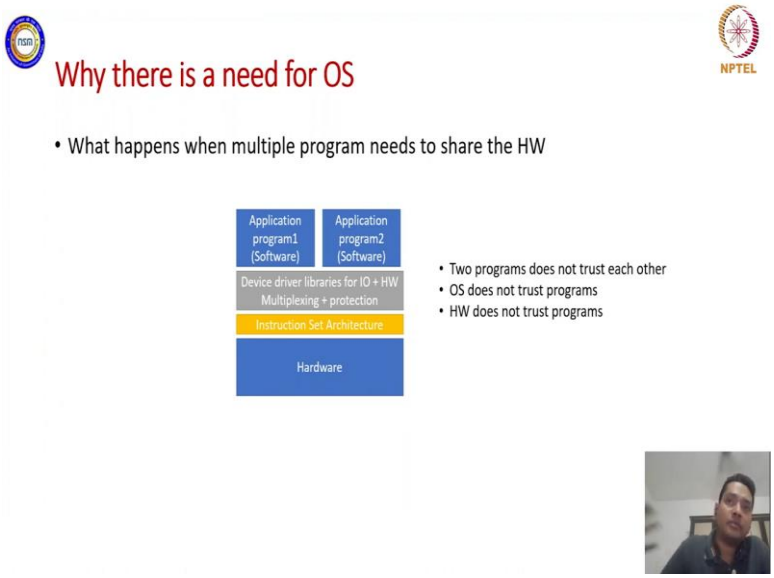
So, if you want to access the memory or store some files or maybe do some operations in the CPU, so all these things will be controlled by this interface called instruction set architecture. Now your application program you have on top of the instruction set architecture. So, if you can access the instruction set architecture then why you will need the operating system right.

So, if you see that all these instructions set architecture can be abstracted into some libraries and those are called the device drivers. Because all the programs that you will run in this underlying hardware, will not be able to talk to all the systems at one time right. So, basically let us say you have program 1 and program 1 only needs, let us say needs to talk to CPU or maybe only memory and CPU.

And program 2 needs to access to your network card and let us say some IO devices and as well as the CPU. So, there are drivers for different components that are underlying. And you have abstracted them, so for to control them you have several sets of instruction set architecture. And then we have abstracted those into libraries driver libraries, for these IO and other devices to have access. Now, you can use these libraries inside your application program.

So, that you do not need to write instruction set architecture for all the accesses and privileges that you want for. So, now you can write these libraries and you can take the help of these libraries and you can access them from the program, you can run them. So, this is kind of the primitive OS that is coming in between the application program and your ISA interface right.

(Refer Slide Time: 06:09)



Now, what if you want to run several programs. Let us say program 1, program 2, concurrent. That means, at the same time you want to run several programs. So, that means several programs need now to access the CPU the underlying hardware now they need some multiplexing. Because if you want to run several programs concurrently; that means, underlying hardware will be multiplex between this between these programs or software codes or user applications.

So, now what we are doing here? We are sharing the resources that are available in underlying hardware through this hardware multiplexing. And we are giving that illusion to these programs that they are accessing the entire resources that are available inside the hardware. Now you can see that we are evolving from this device libraries for IO then order multiplexing.

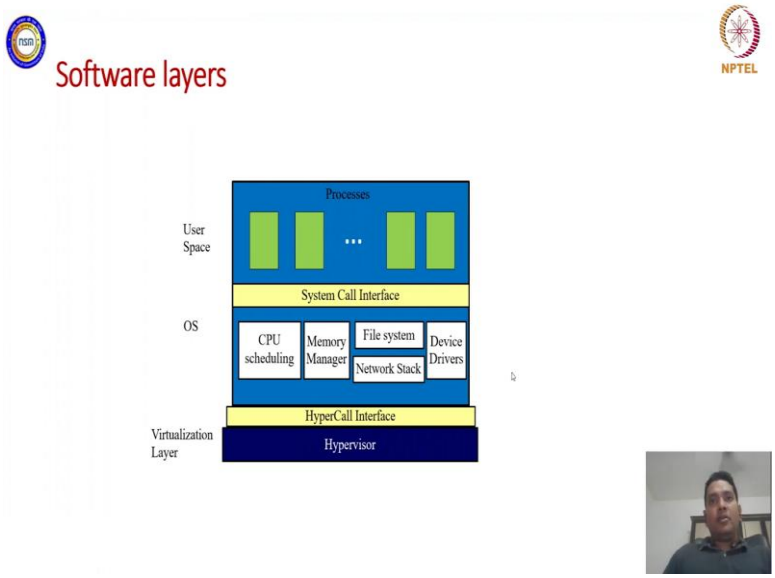
And then you need some protection as well, when multiple programs you have allowed to run on top of your instruction set architecture and your operating system which we are talking about here. There you need to take care of several levels of thread models. So, like two programs which are running together they might not trust each other operating system also might not trust the programs that are running.

So; that means, that the programs itself which are the user level applications that are running on top of your OS, they can try to potentially put some malicious instructions inside your OS ok. And hardware also cannot have the full trust over the programs, but

because you can have the memory access you can you the program user level program can access the privileged sections of memory or so on and so forth. So, different levels of protections or thread models you need to take care.

And all these protection models if you integrate into your device private libraries plus your scheduling, multiplexing your hardware resources and the protection model. Then you have kind of an encapsulated version of the modern operating system. So, this is where the idea of this piece of software lies, because we have the application programs, the operating systems, instruction set architecture. Interface then the underlying hardware which is essentially running all the instructions, that are coming from the operating systems to be done ok.

(Refer Slide Time: 09:25)



Now, if you see the software layers ok. So, we are talking about user level programs we are talking about operating systems which has several libraries and other protection modules, the multiplexer or scheduling modules right. So, all these software are lying in layer wise inside your system. So, you have your user space where all these processes will be running, so processes means the user applications. Then you have the OS, so which has this scheduling memory management file system network stack device drivers and so on.

Now, in the between so user space, so all these processes will talk to the OS through this system call interface. So, these interfaces are very much necessary to understand like

where they are placed, to get a very comprehensive idea about the virtualization that we are trying to talk in the lecture.

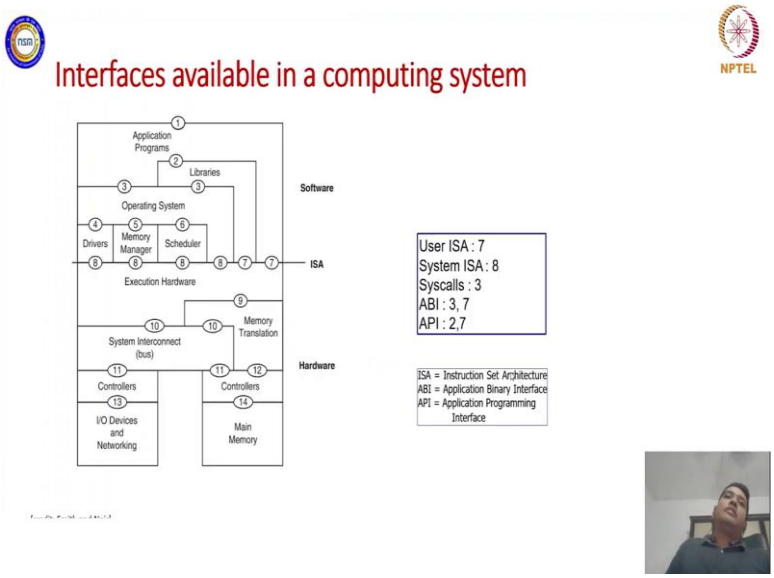
So, now, system call interface. So, basically the privilege instructions like let us say one process wants to access some memory or maybe file system, which is residing in your hard disk or your network storage. So, process will invoke system calls to this operating system which is underlying.

And then operating system will convert them into instruction set architecture, which has several components as user components as well as system components that we will see next, but instruction set architecture and that will go eventually through the hardware underlying hardware. But increasingly in this stack one more layer is getting added which is called as hypervisor or which is widely known as the virtualization layer. So, with hypervisor is also another a piece of software. So, you can see the layer position, so it is actually lying underneath the operating system.

And most of the time hypervisors want to or try to stay invisible or very very transparent to the operating system. But if it wants to show itself before the operating system, then it will have this hyper call interface as the mid interface between the operating system and hypervisor. But most of the times operating system will not know that there is one hypervisor layer underneath it.

But of course, we will see several levels of usage of hypervisors and how we are virtualizing the entire system, but yeah. So, basically this is the software stack that we are talking about and several level of interfaces, system call interface, lying between your user space and OS and hyper call interface which is lying around between your operating system and hypervisor ok.

(Refer Slide Time: 12:54)



So, next talking about interfaces, it is now important to know what is this system level or system call interface and some other interfaces also we need to know to get our knowledge around the virtualization layers because virtualization will be happening in several layers of our computing system.

And that is why we need to know several interfaces that are present inside our computation inter compute system, computing system in different positions. Because when we are talking about the virtualization, essentially what we are trying to do is that we are trying to virtualize the interfaces.

So, let us look at the interfaces that are available in the computer system. So, as you can see here is the ISA the software stack is on top of this. So, basically you have operating system here I will just enable the laser pointer for better visibility. So, here you can see that ISA is here in between and beneath that we have the entire hardware stack. So, you have execution hardware memory translation unit system interconnect bus which is connecting to your controllers main memory and IO devices and controllers.

And on top of that you have this interface called ISA; ISA has this two sets of instructions as we call because we have user level ISA which is 7 labelled as 7 and system ISA which is labelled as 8. So, system ISA will be mostly invoked by the operating systems as the privileged instructions of the privilege calls from the instructions to operating systems through this instruction set architecture.

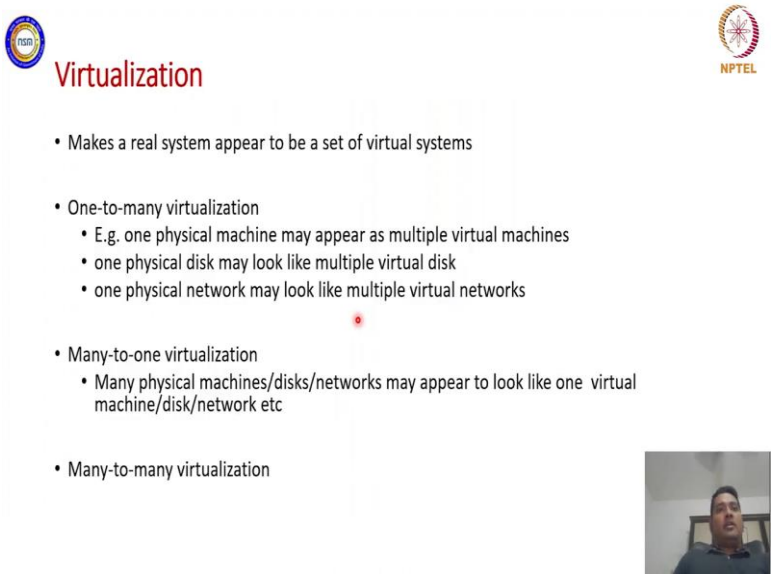
Now, on top of operating system we are having these libraries, now these libraries are essentially encapsulating the system calls and all. And these will give one interface to your application program as this two. So, like these libraries and application programs are interacting with this interface two interface.

So, we will talk about that and an application program also can have the user instructions and can have system call instructions, that are directly directed to your operating system. So, now application programs have two sections open you see here, so 7 is open and 3 is open to your operating system.

So, application programs are making system calls to your operating system to get privileged instructions to be executed and application programs directly talking to your hardware through the user ISA ok, so two sections is 3 and 7. Now 3 and 7 if you combine them you will have this Application Binary Interface or ABI widely known as ABI.

And if you combine this 2 and 7 interface which is between the library the API mostly. So, API will be the view of the libraries and associated with the users ISA ok. So, now, these ABI and API is very important to understand. So, you see this interface 2 and 7 and 3 and 7 right. So, these two interface are essentially you see here application is talking to your operating system and hardware to these two interfaces, either ABI or API. Now ok so.

(Refer Slide Time: 16:54)



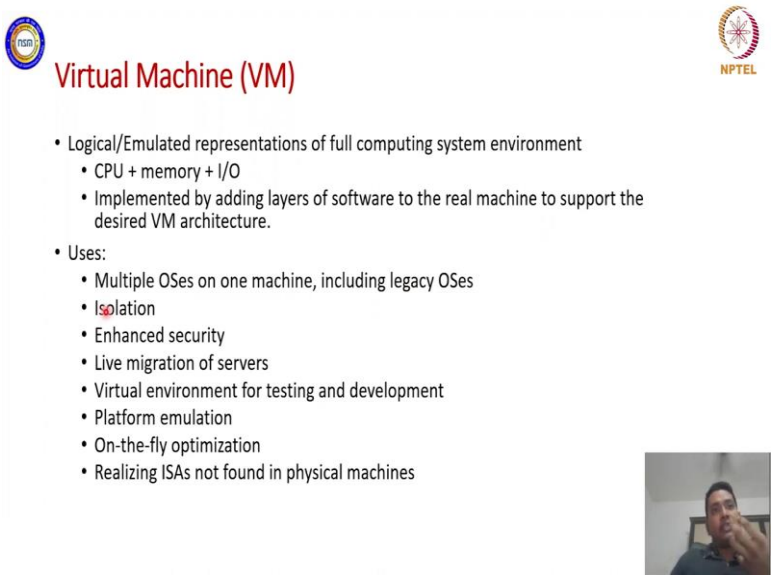
Now we will talk about the virtualization. So, virtualization makes a real system appear to be a set of systems which are virtualized, but they will appear as real system to the operating software. So, basically let us say you have one piece of machine which you want to virtualize. So that means, you can have one-to-many virtualization, so one physical machine can appear multiple machines, multiple virtual machines.

You can have your storage also virtualized, so basically if you have learnt about virtual memory and paging and so on, so that time you have read about virtual memories. So, basically you have one physical disk or physical memory and you want it look like multiple virtual disks right and also the network also may look like as multiple virtual networks.

So, this is from one computing system point of view. So, all the components in your computing system you can virtualize. So, basically you are virtualizing your computing system one to many and your storage unit one to many and your network interface one to many and so on. Now you can have many to one virtualization also.

So, many physical machines which are networked together and they may appear you can also emulate them to look like one virtual machine altogether ok. So, these two concepts will be very instrumental to understand the concept of cloud computing ok. And then you have many to many virtualization which you can just map however you want.

(Refer Slide Time: 19:05)



So, we are talking about virtualization. So, basically first we will look at the virtual machines what way we can or different ways we can virtualize. So, we want to virtualize, so basically when we are talking about virtual machines then the CPU plus memory, the IO the entire computing system that we want to emulate as a full computing environment and we want to virtualize it right.

And implementing by adding layers of software to this real machine to support your desired virtual machines that will be the representation of your virtual machine. So, we will discuss about that broadly like in different levels of computation how you can in your computing system how you will virtualize, but; Let us get motivated, why we will need this virtual machines creative in the first place right.

So, one of the use cases will be you can run multiple OSes in one machine including your legacy OS which is native OS that is already done right. We can have isolation because, let us say several processes you want to isolate them; that means, the underlying resources they are sharing if they are they are executing in one machine. But if you want to virtualize them and if you want to have greater isolation between these processes that are running then you cannot do using these virtual machines.

So, better I isolation I mean better segregation or boundary between the resources they are sharing right or how they are talking to each other. Now you can have enhanced security also because, let us say one user program is malicious and you want to test it whether the user program is malicious or not.

If you are trying on your native OS then if that is malicious then entire OS can get corrupted, but if you want to try them on your. Let us say virtual machine then if the piece of software is a malicious software, then only the operating system or the guest OS will get infected and you can throw it out and renew one new virtual machine right.

So, that way you can have enhanced security as well as you can have better boundary between different processes that will be running on different virtual machines that we will see that will be the one of the important factor which will create the convention of containers and so on. So, that we will see.

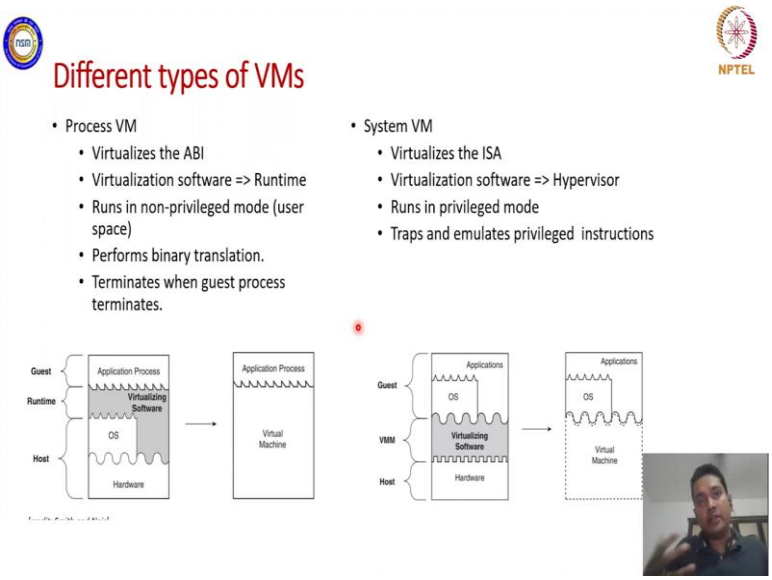
Live migration of servers, so without shutting down your PC you can actually or physical machine you can migrate your servers from one virtual machine to one another virtual

machine. You can create virtual environment for your testing development of your processes or your user applications.

You can emulate your platform, let us say you have created one new ISA which you will be running on onto your newer system or newer CPU right. Now you do not have yet the hardware, but you want to emulate them for your running. So, basically you want to test how we perform the new ISA.

Now at that time you can virtualize and you can test in your virtual machine for this targeted ISA new targeted ISA. You can do on the fly optimization as well, your user program will run and on the fly you can optimize by virtualizing. So, all these things you will see realizing ISAs which is not found in physical machines that, I just talked about in one such users use cases of platform emulation ok. So, all these are different use cases and motivation for getting introduced to the virtual machines or VMs right.

(Refer Slide Time: 23:28)



So, let us talk about different types of VMs, so that the ideas will be clarified like where we can virtualize and what are the levels of virtualization we can do and what are the advantages and disadvantages correct. So, we talked about hardware and on top of that we have OS and OS and hardware in between we have this interface called as ISA right.

And let us see you have this hardware x86 hardware then your ISAs x86 ISA. Then you have this OS and if your application will be running or process will be running then the

ABI which is this system call interface and your ISA user ISA right. So, this was the boundary for your application process and hardware.

Now you want to run applications for let us say virtualization virtualizing software. Let us say this is one new ISA you want to try right and there you want to virtualize this software. So, from this ISA to this ISA conversion you introduced one new virtualization software which will emulate that, conversion of this ISA from this ISA.

Such as let us say JVM right. So, a Java Virtual Machine you write your programs your Java programs which will be actually the compiled to your byte codes and which will run only on Java virtual machines. So, Java Virtual Machine is the virtual machine which will transform the ISA byte code to your x86 ISA and it can run it right.

So now, so what is the advantage of this process level virtual machine. So, what we are talking about this is the process VMs the we are virtualizing the ABI. So, ABI was the interface combination of the system call interface and the ISA user ISA right, 2 and 7 you remember that from the previous slides. Now virtualizing software or process level software is essentially we are virtualizing the ABI.

Now new ISA ABI this translation it will be done by this virtualizing software and it will perform the binary translation and it will. So, this is the advantage of this virtualizing the process virtualization or process level virtualization. Where if we want to run this application process into another system, let us say some arm system right or maybe a AMD system or maybe so AMD also uses x86 ISA.

But if you want to use any other system like Power PC or any other system then you can have this virtualizing software install and you do not need to change anything inside your application process, it will directly run on the byte code or byte code interface that will be provided by these virtualizing software, that byte code is just an example in this case.

Now application process is just seeing this world of this ISA that is being or this interface provided by this virtual machine. This is no longer dependent on the application binary interface, now system virtual machines are virtual machines which virtualizes the ISA itself. So, this process virtual which is virtualizing the ABI here in the system level or system virtual machines we are virtualizing the ISA itself.

Now, ISA means this was the underlying interface between this OS and hardware, so basically this was the ISA now we are virtualizing this. So, basically this virtualizing software will run in privilege mode, so basically this will query this hardware for the privileged instructions. And so, if you see that hardware this interface between the virtual software; that means, the VM system level VM or system VM and on top of that you have operating system and applications.

So, operating system and application is looking at this interface, which is the system interface and let us say use the ISA right. So, this is the ABI for your application and this is your system interface for your or ISA interface for your OS. Now what it is doing is that OS will do the privileged instructions as it was doing to its native hardware, but now since virtualizing software it is translating the ISA to another ISA interface you see.

Let us say this is x86 ISA interface and let us say this is arm ISA interface right. Now, now this system level virtualizing software what is doing is that it is translating this privileged modes and converting the ISA from this interface to this interface. OS does not know that because OS is looking at this interface and that is completely fine to OS because, that underlying hardware was supposed to be the x86 ISA right.

Now what virtualizing software it is doing is that it is. So, whenever OS is sending or assigning the privileged instructions, it is trapping those instructions and it is emulating them in the underlying hardware. And then it is returning the return calls to the operating system, so this is how it is doing.

Of course, there is will be some penalize of resources because this will be using several because this trap and emulate will cost some hardware and resources. But the main important thing is that you can now have this OS looked this world of virtual machine. It is completely unaware or unaware of the underlying hardware that is interfaced with this; that is right.