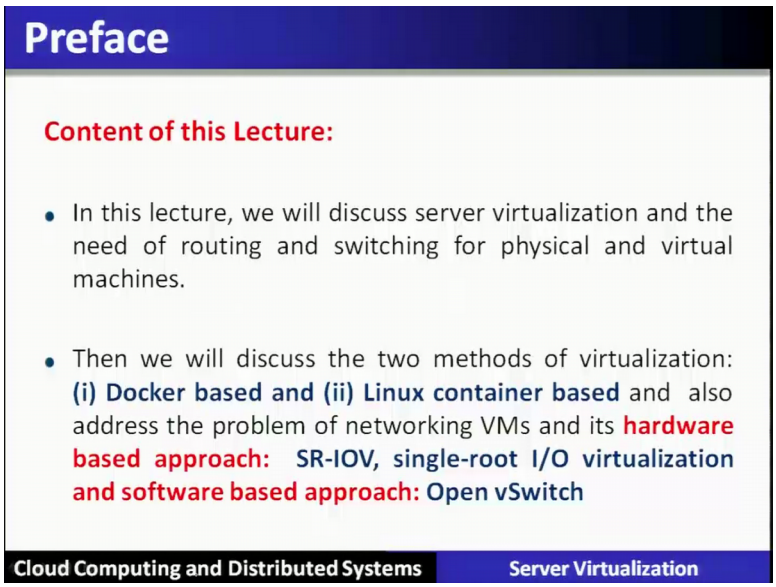
Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture - 04
Server Virtualization

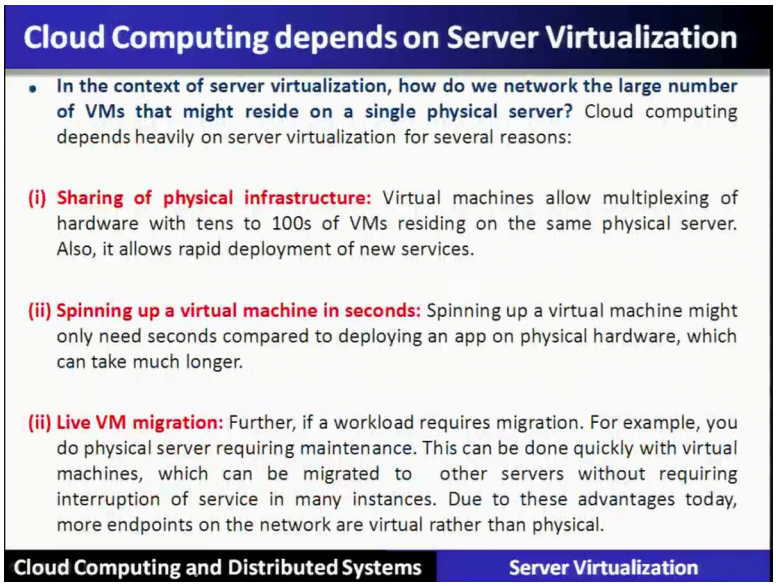
Server Virtualization; preface content of this lecture.

(Refer Slide Time: 00:18)



In this lecture, we will discuss server virtualization and the need of routing switching for physical and virtual machines. Also, we will see 2 other methods of virtualization that is the Docker based method and Linux container based methods. We will also see the problems of networking of VMs in this context we will cover 2 different ways one is hardware based approach that is SR IOV, the other one is software based approach that is using open vSwitch.

(Refer Slide Time: 01:01)

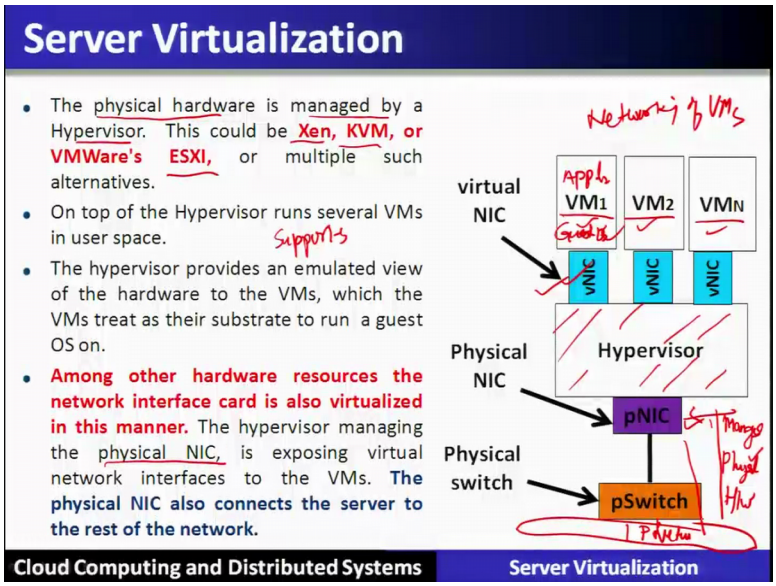


The cloud computing heavily depends on the server virtualization. So server virtualization we will see how do we network the large number of VMs which are running on a single server machine. So, the cloud computing heavily depends on the server virtualization because of these several reasons, the first reason is sharing of physical infrastructure. So, virtual machine allows multiplexing of hardware to that hundreds of VMs residing the same physical server. So, therefore if such a large number of VMs are spanning on a same physical machine. So, now the issue comes of it is rapid deployment whenever a new services are coming.

So, the spinning of virtual machines need to be done in a fraction of seconds, so spinning of virtual machines might only needs some fraction of seconds compared to the deploying an application on a physical hardware, so this is another reason of the server virtualization.

Third most important reason is the maintenance, whenever there is a maintenance in the physical server then all the VMs need to be migrated to some other server that is called live VM migration. So, this will allow that the services which are running through the virtual machines are not disrupted in a state they will be migrated that is called live VM migration. So, due to this feature the server virtualization is also going to be very important in realizing the cloud computing or in realizing the today's cloud.

(Refer Slide Time: 03:23)



Let us see what is a server virtualization? Now we see that in this picture that hypervisor manages the physical hardware, so this is the hypervisor and it manages. Now this particular hypervisor could be Xen hypervisor or it could be Linux kernel based virtual machine that is KVM hypervisor or VMwares ESXI hypervisor or many other hypervisors are available.

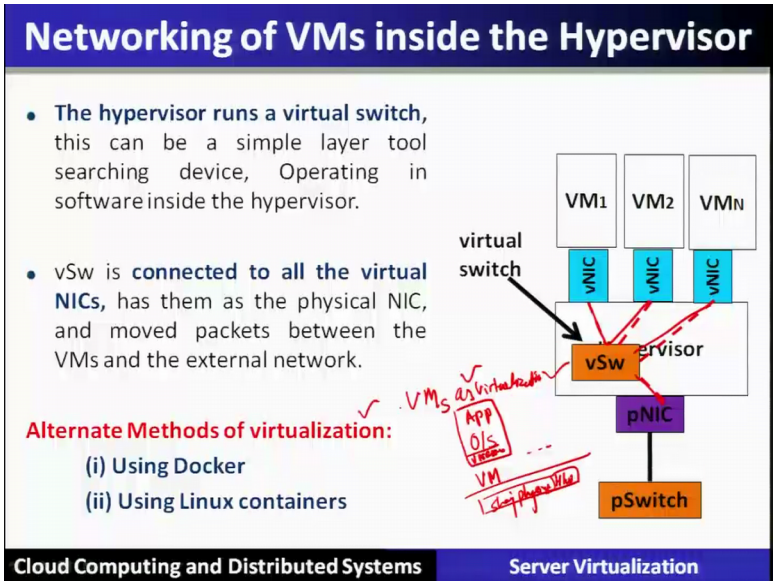
So, this hypervisor is managing the hardware physical hardware resources using this managed management of physical hardware resources it also support several virtual machines. So, it supports several virtual machines which runs on top of these hypervisor. So, this hypervisor will provide an emulated view of the hardware to these virtual machines, which the virtual machines treat as their substrate to run on the guest operating systems. So, these virtual machines they have the guest operating system and this is the application. So, this guest operating system will have the view as if it is owning all the physical resources to run it is application and that is being supported by the hypervisors.

Now, among other hardware resources the network interface card is also virtualized in this particular manner. So, these VMs are need to be networked and thus physical network interface card NICs also need to be virtualized. So, that all the VMs can be networked together, so the hypervisor which is managing the physical network interface card is exposing the virtual network interface card that is called VNICs virtual NICs to the VMs. So, every VM will be having a virtual network interface card and this allows

the networking of all the VMs. So, the physical NIC also connects the servers this server to the rest of the world.

So, in this particular server virtualization we see that how the networking of these VMs are to be done through the virtual NICs, which basically is managed by the hypervisor which is managing the physical NICs and it will provide the support to the virtual NICs which will intern network all the VMs. You know that NICs are responsible to connect the servers to the outside world, so this way all the VMs also will get the connectivity in this form.

(Refer Slide Time: 07:33)

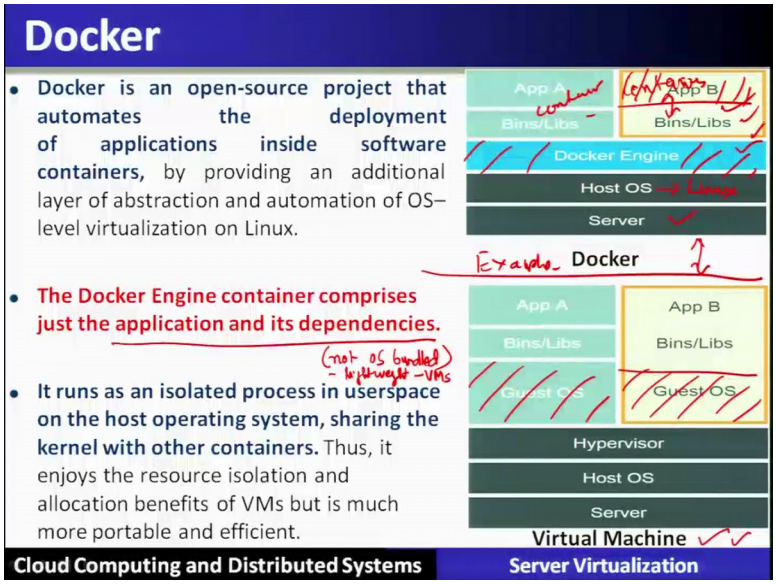


So, that working of VMs inside hypervisors is done we will see the more detail of it. The hypervisors runs a virtual switch, so inside the hypervisor you can see here the virtual switch is represented. So, the hypervisor runs the virtual switch and this can be a simple layer tools of a switching device and the operating system in the software inside the hypervisor. So, virtual switch is connected this virtual switch is connected to all the virtual NICs and also it is connected to the physical NIC. So, virtual switch will basically which is managed by the or which runs inside the hypervisor provides the connectivities to the virtual NICs by managing the physical NICs.

So, whenever a packet of a particular VM comes through the virtual NICs. So, it is the virtual switch which will basically use the physical NIC and therefore these packets are routed to the external world. Now we will see this particular aspect in more details, but

before that let us see some more alternative methods of virtualization. We have seen the virtual machines as the virtualization, where in every application is bundled with it is corresponding operating system and it will have the virtual resources this is called a virtual machine. This allows the sharing of the physical hardware resources or let us say the server this type of virtualization we have already seen. Now, there is an alternative to it using the Docker and the Linux container let us see these further approaches and then we will come back again to the virtual switch.

(Refer Slide Time: 10:24)



So, Docker is a method for virtualization as I told you. So, Docker is an open source project that automates the deployment of application inside the software container by providing additional layer of abstraction and automation of OS level virtualization on the Linux. Take this example on the figure here there is a Docker engine on top of it, it runs the containers this is container this is also container. This Docker engine is just above the host operating system this is nothing, but a Linux kind of operating system runs over the server.

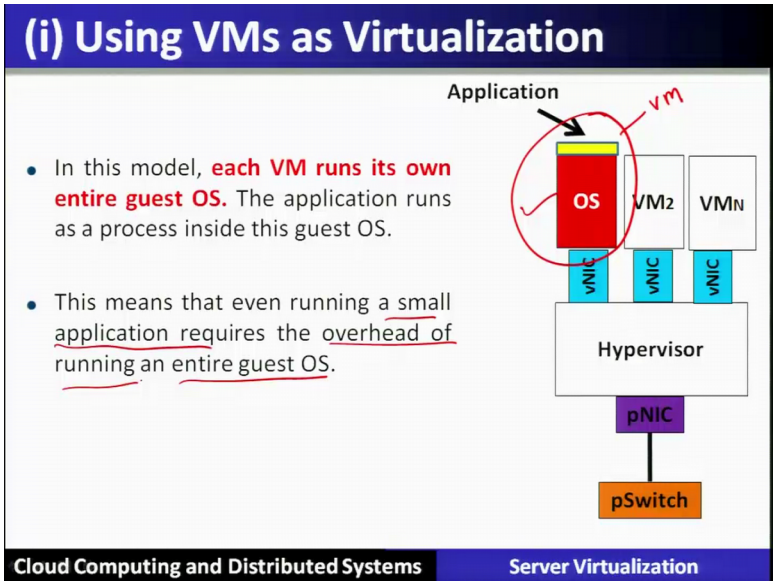
So, the Docker engine here it basically figures out in the picture contains Docker engine container comprises just the application and it is dependencies. So, unlike in virtualization we have seen the application with it is corresponding operating system bundled together, but here the application corresponding application and it is dependencies. Dependencies means the binaries and the libraries which require to run

that application not the entire operating system, so it is a some part of which was used earlier and this together is called the container.

So, now here the application and it is corresponding dependencies they are called as the container, we have shown the container over here and this is the simple virtual machine which also has the operating system. So, you can see the comparison between the Docker container based virtualization and the simple virtual machine where the entire guest operating system is also bundled with the application. So, the D ocker engine container comprises just an application and it is dependencies not; so that means, it is not having the OS bundled. So, hence it is a lightweight in comparison to the virtual machines based virtualization.

Now, it runs as an isolated process in the user space that you see because it is running above the operating system. So, it runs as a user space on the host operating system sharing the kernel and other container, thus it enjoys the resource isolation and allocation benefits of the virtual machine, but is much more portable and efficient why because it is managed by the Linux operating system all the containers.

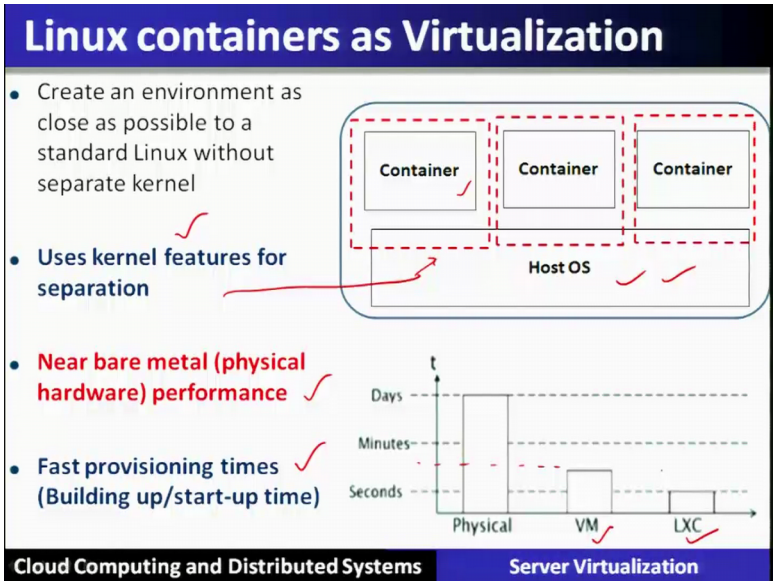
(Refer Slide Time: 13:40)



So, let us see without Docker and container that is simple virtual machine based virtualization, in this model you see that each VM runs it is own guest operating system. So, just see that the application bundled with the operating system with the virtual NIC is called virtual machine. So, this means that even running a small application it requires to

be bundled with the entire guest operating system. So, it becomes a big overhead if the applications are a little or quite small, so this is the drawback of virtualization if it is done using virtual machines.

(Refer Slide Time: 14:38)



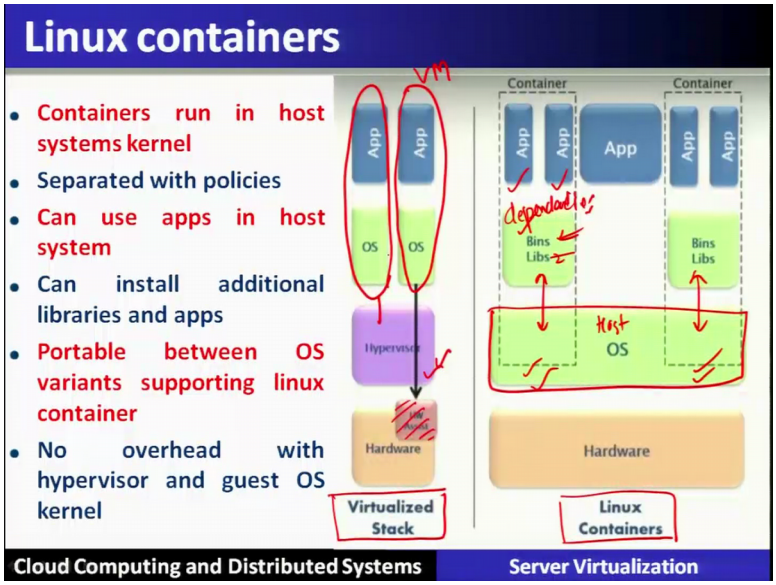
On other hand if Linux containers are used as the virtualization, then you see the container is different than the host operating system and container does not contain the operating system. So, this will create a environment as close as possible to the standard Linux without the separate kernel. So, there is a same operate host operating system which will now manage or which will give the support to all the different containers.

So, it uses the concept of Linux kernel based features for the separation or the isolation which are shown over here as dotted blocks which provides an separation and isolation. So, this will ensure that the container the applications which are there inside container running they will run having the performance near bare metal performance that is very fast.

Also it has the advantage that it has a fast provisioning times that is since it is a light weighted container do not contain the operating systems within, but it uses the host operating system. So, the provisioning time that is building up or the startup time is also very little, in this particular case take the comparison between the virtual machine and the Linux based containers.

So, the fraction of seconds here it is much more than or near to the minutes as for as a VMs are concerned. Therefore, the containers as the virtualization is one of the important features of virtualization, which is used in the server virtualization to support large number of applications running in the same server, compared to the to the VMs which will not be that many number of applications compared to the same capacity.

(Refer Slide Time: 17:08)



So, Linux containers run in the host system kernel and it is also having the separated with the policies; that means, every container it is own policies and can install it is additional libraries and binaries which are required to run this app and the same host operating system will support running these containers. So, they are portable between operating system variants supporting the Linux containers and there is no overhead with the hypervisor and the guest operating system kernel.

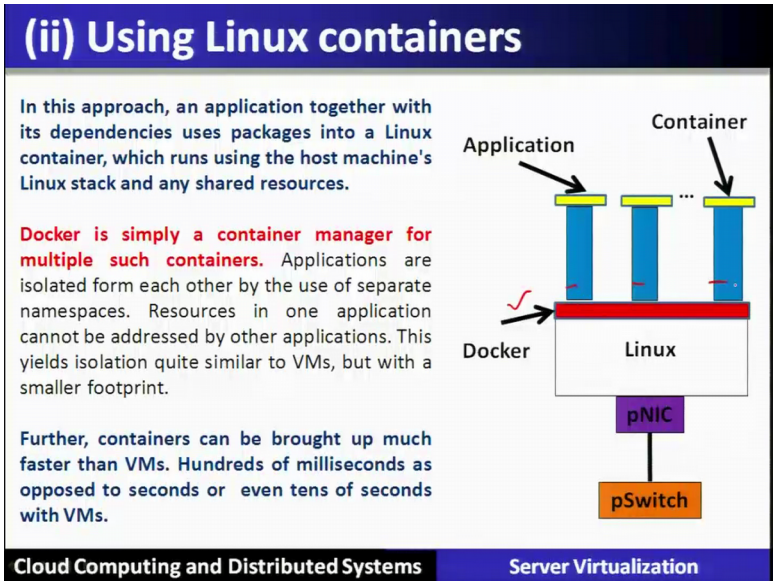
So, you can see here this is the example of a virtualized stack, in the virtualized stack there is a hypervisor and every application with it is own operating system is called virtual machine and there is a special hardware required to support to support the hypervisor to run these virtual machines.

On the other hand you can see the right side this Linux containers, this Linux containers are light weighted they do not have bundled or the containers running their applications requires only the dependencies within it. That is the binaries and the libraries which

required by the application during the runtime and together is called container and this is going to be used using the host operating system.

So, the operating system is basically one operating that is the host operating system will be meant will be supporting all the containers in contrast to every application having the separate operating system.

(Refer Slide Time: 19:24)



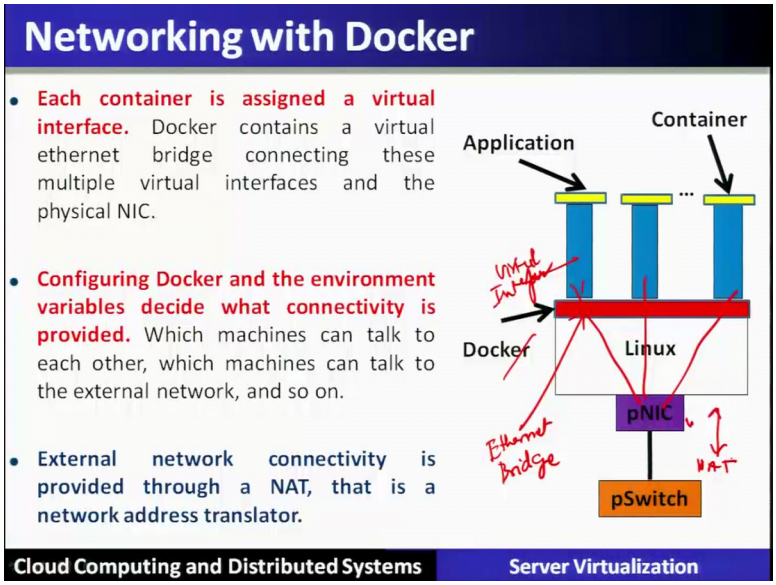
Now, coming the more detail if when using the Linux container as the virtualization. Now in this approach an application together with it is dependencies uses packages into the Linux containers that we have already seen which runs using the host machines Linux stack and any shared resources.

So, the Docker is so that was the container; what is the Docker? So, Docker is simply the container manager. So, it manages all the containers running on that particular server such that it has to provide the applications with the isolation or the separations by way of giving separate namespaces. So, resources in one application cannot be addressed by the other applications why because, they are in a different namespace and which is being given or which is being supported by the Docker.

So, this yields the isolation quite similar to the virtual machines but with a smaller footprint. So, without having the separate operating system yet it is able to provide all support to the containers and that is being managed by the Docker. Further containers

can be brought up much faster than VMs 100 of milliseconds as opposed to the seconds or even 10 of seconds with the virtual machines and that is why the Linux containers and the Dockers are becoming popular way of virtualization.

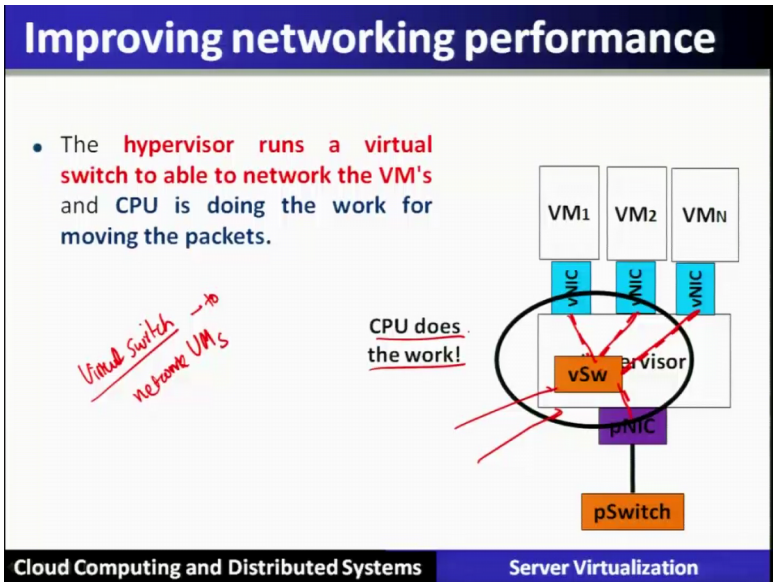
(Refer Slide Time: 21:11)



Now, comes the networking so each container is assigned about a virtual interface and Docker contains the virtual ethernet bridge. So, Docker contains a ethernet bridge connecting these multiple virtual interfaces, all these virtual interfaces are to be connected to the physical interface using this particular bridge which is been provided by the Docker.

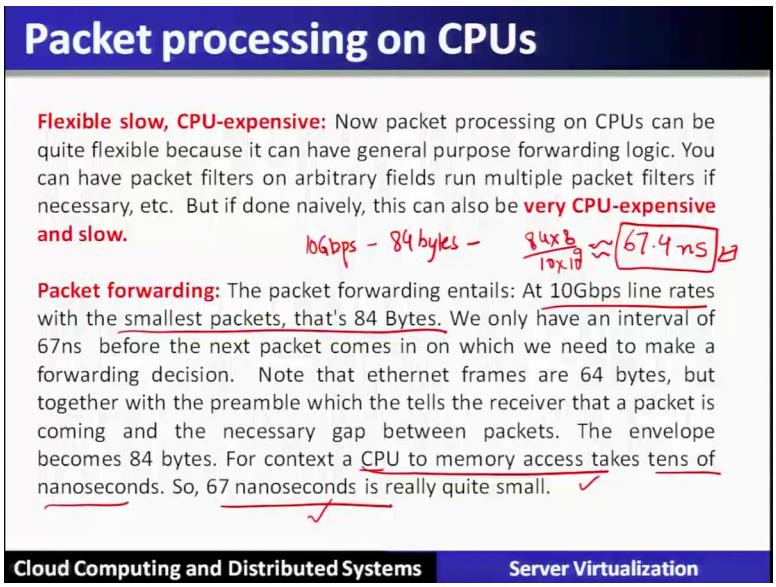
Now, configuring the Docker and the environment variable we will decide what kind of connectivity is going to be provided to the container that is which machine can talk to each other which machine can talk to the external network and so on. So, external network connectivity external network connectivity is provided through a NAT that is called the network address translator.

(Refer Slide Time: 22:40)



Now, let us see the performance of this kind of networking and how it is going to be affected or improved. So, the hypervisor runs a virtual switch that we have already seen, but we will go in the great detail. So, the hypervisor runs a virtual switch which is able to network the VMs, so this is the location of virtual switch which is basically running in the hypervisor and this will give the networking to the virtual NICs or to the VMs while managing the physical NIC. Now, who will be responsible for moving the packets, obviously it will be the CPU which will be responsible to move the packets.

(Refer Slide Time: 24:15)



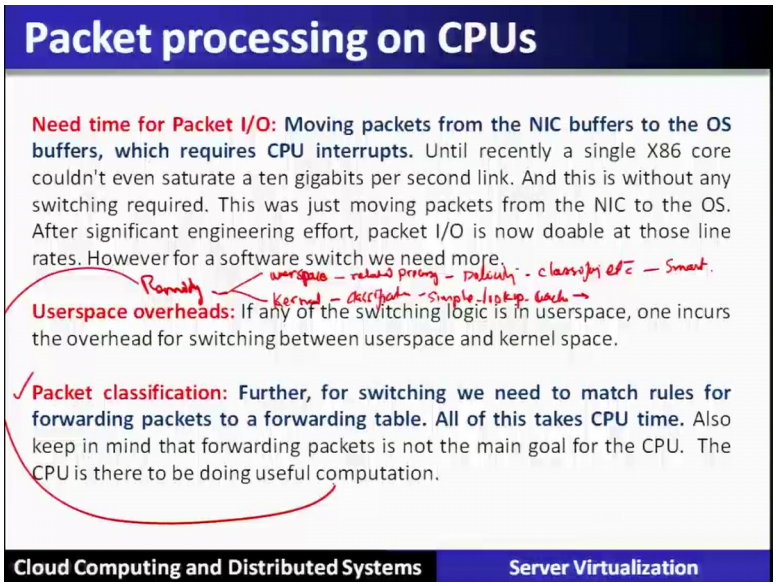
Here, in this case; so the packet processing will be done here in the CPU. Now CPU will be now busy doing the packet processing now the goal should be that CPU should be relieved out of this packet processing. So that means, very less attention of CPU should be required here in the packet processing, so that CPU should do more of computation job.

So, let us see all these aspects now this particular approach is flexible slow and also CPU expensive. Now the packet processing is done by the CPU it acts the flexibility because it can have the general purpose forwarding logic in place. So that means, you can have the packet filters on arbitrary fields which will run multiple packet filters. If necessary therefore, lot of flexibility as far as packet processing is concerned is done, but with the cost of this flexibility is now born by the CPU expensive operations.

So, hence the CPU will be expensive most of the time CPU we will be doing the packet processing and also it becomes slow. Now another aspect which is called a packet forwarding the packet forwarding entails let us take an example, then we will understand the intricacies of packet forwarding. Now with the ten gigabits per second line rates and the size of the smallest packet let us assume of 84 bytes. Let us see what is the intervals of time to process the packets by the CPU. So if you find out how much time is required to process the 84 bytes packet. So, you will come out to be 84 into 8 divided by 10 that comes out to be this 10 GBPs that comes out to be 67.4 nanosecond. So that means, 64 nanoseconds is required by the CPU to process the packet. Now let us see that what is the time which CPU take for the memory access that also is in the range of 10 of nanoseconds.

So, most of the time the CPU will be busy reading from the memory these particular packets into the buffer and processing it why because reading from the memory that is I O also takes 10 of nanoseconds and packet processing also takes 67 also in the range of nanoseconds. So, hence packet forwarding is going to be looped into it, why because within a very small gap. That means, how the packet is to be processed and forwarded, so more insight is required so that this packet forwarding has to be dealt properly by the CPU.

(Refer Slide Time: 28:22)



Now, as I told you that it needs the time for packet I O that is moving the packets moving from NIC buffer to the operating system also requires CPU interrupt. Since this requires 10 of nanoseconds and with also interrupt adding to it makes this particular the entire packet processing is quite slow. So, the gap between the packets is very small it is very difficult to process the packet at this speed. So, let us see how this particular problem will be solved if the packet processing is to be done by the CPU.

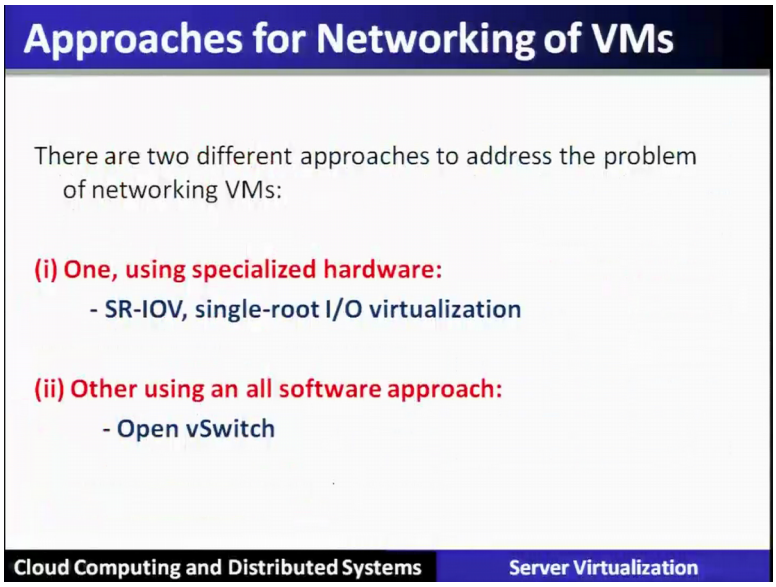
Now, here we can see that most of the tasks if it is done inside the kernel which is not possible, so most of the logic has to be in the user space. So, the use space overhead. So, if any of the switching logic is in the user space 1 occur 1 incurs the overhead for switching between the user space and the kernel space. So, what is basically the remedy that instead of doing everything within by the CPU, so we have to divide this packet processing into 2 parts one is the user space related processing that is called different policies classification etc and this can be incorporated more advanced using a smart way of doing it.

The second is to be done inside the kernel that is the packet classification, this can be the simple packet lookup using cache and therefore this packet processing can be speed up. So, let us see the packet classification further for switching we need to match the rules for forwarding the packet to the forwarding table, all this takes the CPU time also keep in

mind that the forwarding packets is not the main goal of the CPU. So, CPU has to be there doing this other useful computation.

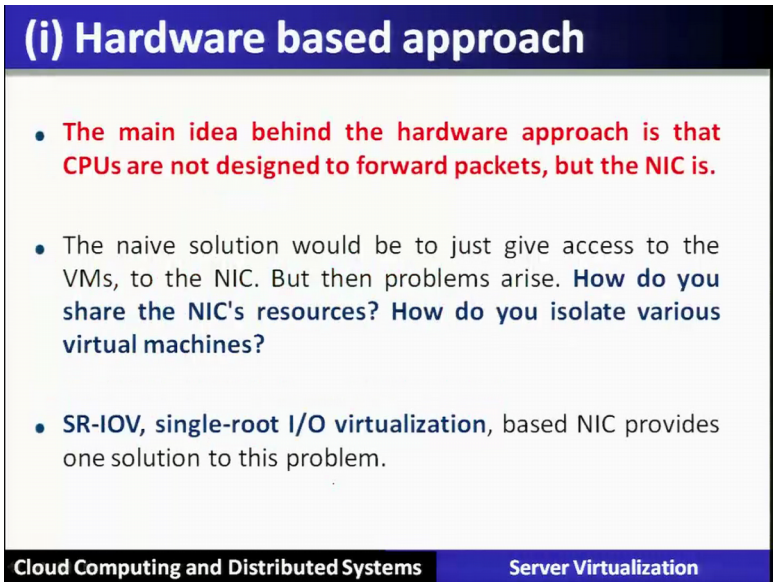
So, this particular problem will be resolved here in this case this the entire computation of a packet processing is divided into 2 parts some part can be shifted to the user space and the remaining part can be there in the CPU. So, can be there in inside the kernel this way the packet processing can be speed up and this problem can be resolved.

(Refer Slide Time: 31:38)



So, there are 2 different approaches to address this problem of networking the virtual machines. As I told you so one is using the specialized hardware which is called SR IOV that is single root I O virtualization and the other part is using the software based approach which is called a open switch. So, these are the networking these are the approaches for networking of VMs.

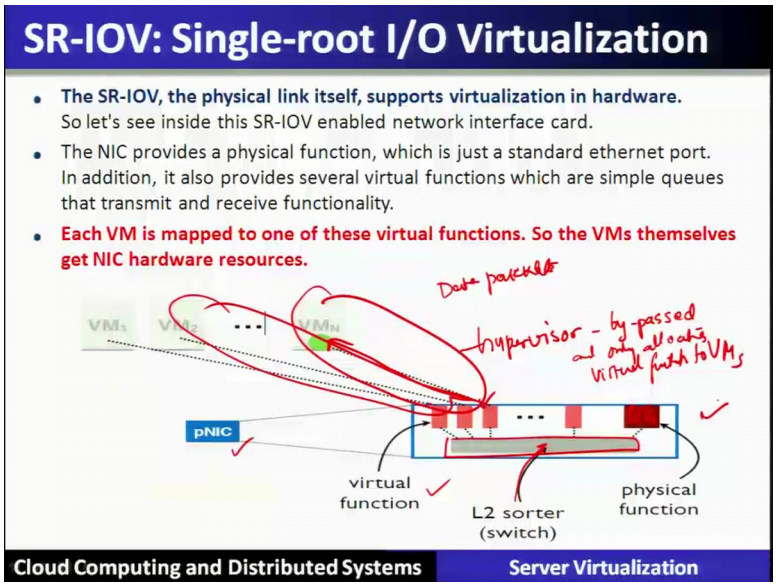
(Refer Slide Time: 32:08)



Let us see in more detail these 2 approaches, so the first one is hardware based approach the main idea behind the hardware based approach is that CPUs are not designed to forward the packets. But NICs are designed for it so the naive solution would be to just give the access to the VMs to the physical NICs, but then the problem arise how do you share the NIC resources. How do you isolate various virtual machines different packets.

So, the solution comes out to be in the form of SR IOV enabled NIC cards. So, SR IOV full form is single root IO virtualization based NIC cards are there which will provide this kind of solution, that it will provide the access to the virtual NIC to the physical NICs access to the VMs to the physical NICs and also ensures the isolation and other properties which are required in the hardware based approach.

(Refer Slide Time: 33:18)



Let us see in more detail about this how SR IOV does provide the access of NICs to the virtual machines. So, SR IOV the physical link itself supports the virtualization in the hardware, so let us see inside the SR IOV enabled network interface card. So, let us see in more detail what the physical NIC comprises of. So, physical NIC comprises of a physical function that is the port and it also provides the virtual functions in the form of buffers which are linked to each virtual machines and these virtual functions intern are connected by an L2 based switch which will basically start the incoming packet to these virtual functions which in turn will be delivered to this virtual machines.

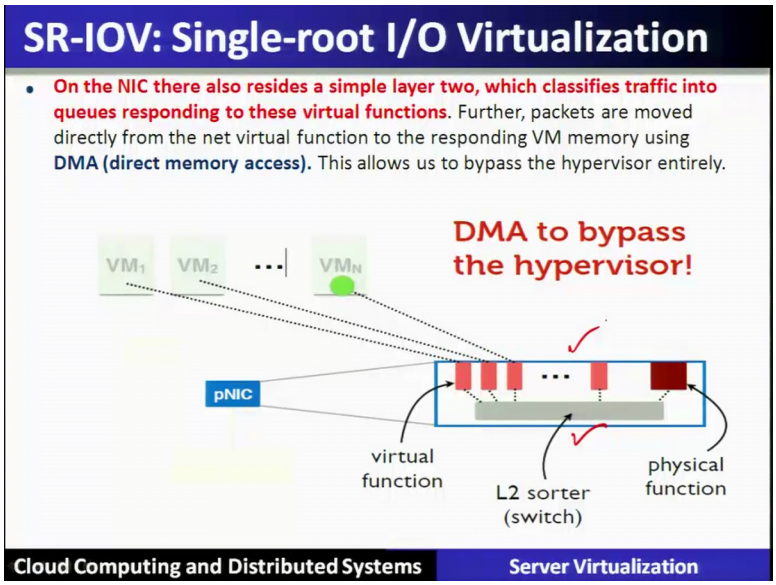
This way this virtual machines can directly be able to access, the network interface cards NICs with the way with the help of virtual functions and hypervisor is by passed and only allocates the virtual functions to the virtual machines. So, this particular mapping is done by the hypervisor, virtual machine and it is corresponding virtual functions. So, this mapping will be done by the hypervisor and then hypervisor will be by passed for the data packets. So, the data packets processing will not be done by the hypervisor it can directly be done by the NICs. So, it becomes a bit faster that is at the wire speed.

So, NICs provides the physical function which is just a standard ethernet port, in addition it also provides several virtual functions which are the simple queues that transmit and receive the functionality or these virtual functions are mapped to the virtual machines.

So, whenever the packets are coming and going from the virtual machine they are basically mapped to a; it is corresponding virtual function.

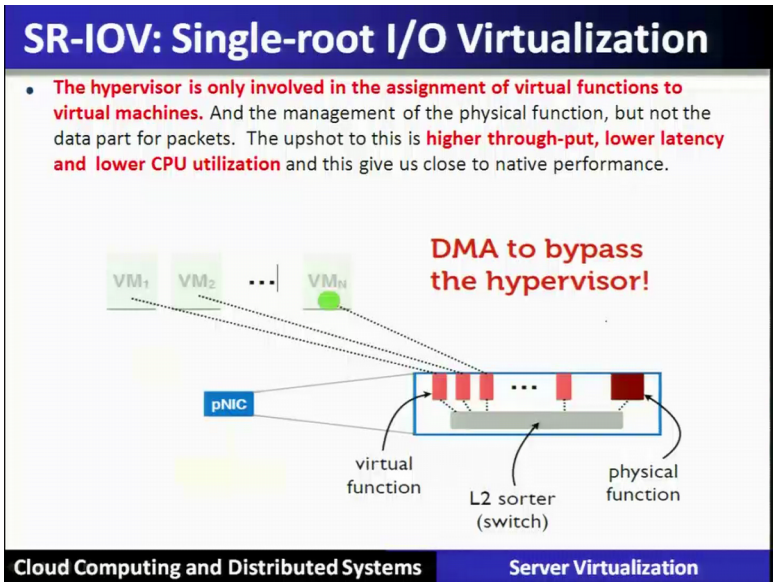
So, each virtual machine is mapped to these virtual functions. So, the virtual machine themselves get the NIC hardware resources directly without the hypervisors intervention. So, hypervisor will only allocate the virtual machines to the virtual functions and then the virtual machine can directly access the physical NICs for packet processing, so you can see that it can work or it can run at the wire speed.

(Refer Slide Time: 36:50)



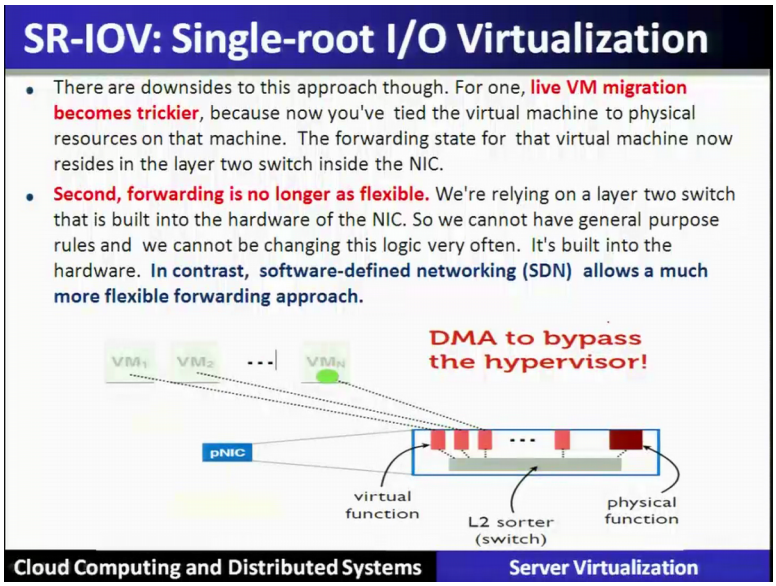
On this, NICs there resides a simple layer 2 switch, which classifies the traffics into the queues corresponding to these virtual functions. So, this is the L2 switch which sorts the packets to the corresponding buffers and which in turn will be mapped to the virtual machines. Further packets are moving directly from the network virtual functions to the corresponding virtual memory using the direct memory access, so this allows to bypass the hypervisor entirely.

(Refer Slide Time: 37:36)



So, the hypervisor is only involved in assignment of virtual functions to the virtual machines and the management of physical functions not the data part of the packets is done by the hypervisor. So, there data packets are directly dealt with the physical NICs and this physical NICs can ensure with the help of L2 switch and virtual functions which will connect or which will network to the virtual machines. So, this may a higher through put will be achieved and latency can be reduced and also the CPU utilization can be will be lowered. So, CPU utilization will be lowered for packet processing; that means, CPU will be free to do it is own computation using SR IOV. So, that is why SR IOV is widely used for networking of VMs.

(Refer Slide Time: 38:46)

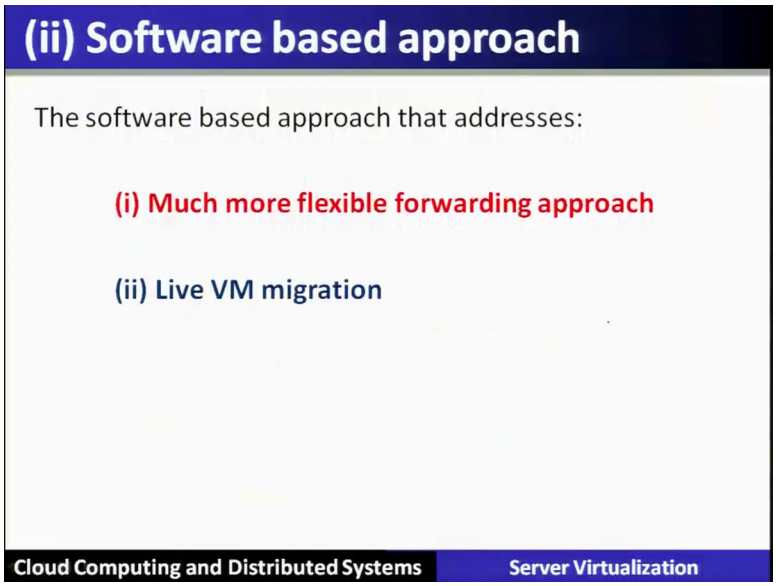


Now, there are downsides to this approach that is when it comes to the live VM migration it becomes trickier, why because all the VMs are now tied up to the physical NICs through the virtual functions and layer 2 switch of the physical NICs, so when they are tied up then the live migration.

That means, migrating from one server to another server will become a problematic, now the question is how this networking or VMs is to be ensured so that it can detach from the physical NICs, so that the live VM migration becomes easier. So, the forwarding state of that virtual machine now resides in the layer 2 switch inside NICs and therefore live VM migration becomes problematic. Second thing is that forwarding is no longer as flexible why because, the forwarding is relying on layer 2 switch that is built into the hardware of NIC, which does not have much smart logic to be in to be dynamically changing.

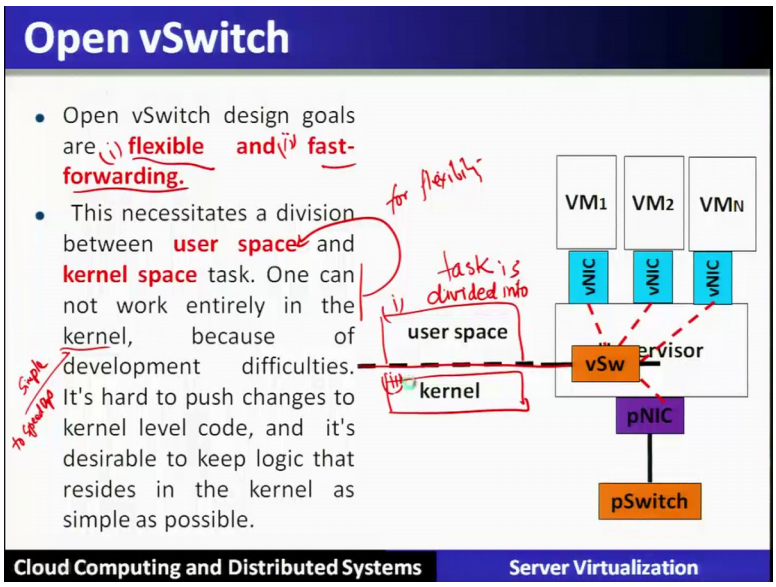
So, we cannot have the general purpose rules and we cannot be changing this logic very often and do dynamically, why because it is built into the hardware therefore, to elevate from these 2 problems a software defined networking allows a much flexible forwarding approach that we will see later on.

(Refer Slide Time: 40:38)



Now another approach which can solve this particular problem these 2 problems; that means, to provide more flexible forwarding approach and also to enable the live migration we will see the software based approach for this particular problem.

(Refer Slide Time: 40:58)



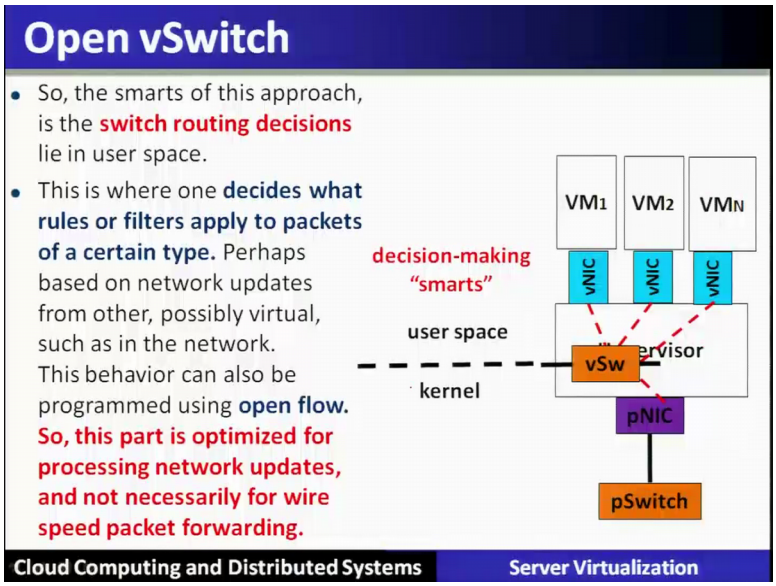
Open vSwitch; open vSwitch design issues are flexible and fast forwarding packet processing. This necessitates a division between the user space and the kernel space task as we have seen in the previous slides, that is we have already built the background to understand this particular division of user space and the kernel space tasks. Because one

cannot work entirely in the kernel space if you want to get the flexibility you have to come out from the kernel space and those flexible rules and filters can be implemented in the user space.

So, if at all if you want to provide the flexibility for that sake such tasks are to be shifted which is one of the goals. So, some of the tasks will be shifted to the user space and the kernel will be relieved with the simple task. So, that the packets forwarding can be speed up now let us see how this particular division between user space and kernel space of this packet forwarding task we have to do.

So, in this particular example here we can see that the open vSwitch, activities are now divided into the user kernel space and user space and the kernel space. So that means, that task is divided into user space and the kernel space. You will see more details what task goes into the user space and what task of virtual switch will fall into the kernel space by putting some of the task in to the user space is to ensure the flexibility. Whereas the relieving those operations from the kernel will increase the fast processing or the fast forwarding, so both these goals is going to be achieved by this particular division of work.

(Refer Slide Time: 43:53)

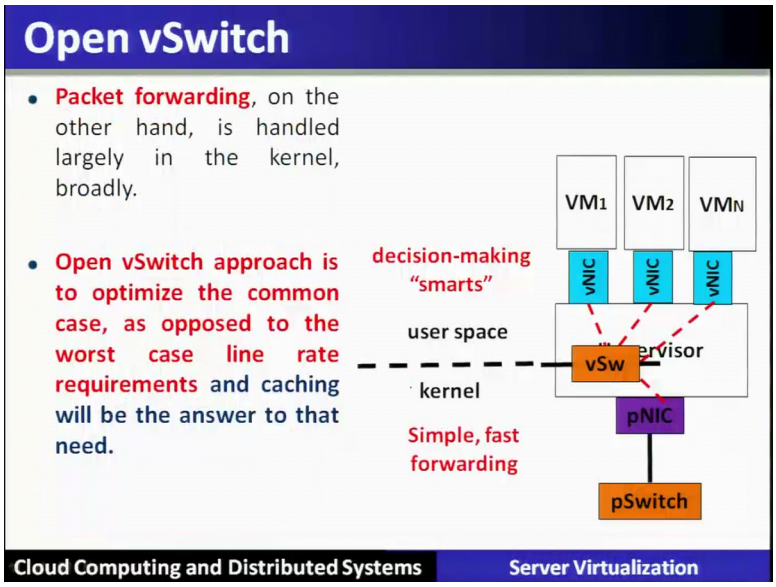


So, the smarts of this approach is the switch routing decisions that will lie into the user space, this is where one decide what rules or the filters applied to the packet of a certain type; perhaps based on the network updates from the other such vSwitch information

which gathers this information and propagates among all other vSwitch. So, that they may learn and may update the rules and the filters, so that it will have the smart decision making of packet processing.

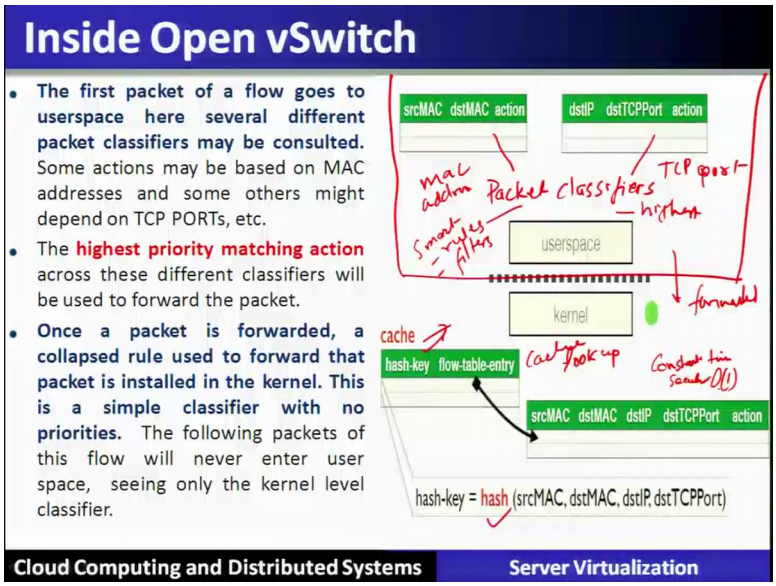
So, this behavior can also be programmed using open flow. So, this part is optimized for processing the network updates and not necessarily for wire speed packet forwarding that is to be taken care by the kernel.

(Refer Slide Time: 44:54)



So, the packet forwarding is handled largely into the kernel, so let us see what the kernel will be doing. So, open vSwitch approach is to optimize the common case as opposed to the worst case line rate requirement. So, basically the alternative for the first packet forwarding is to be done using caching.

(Refer Slide Time: 45:28)



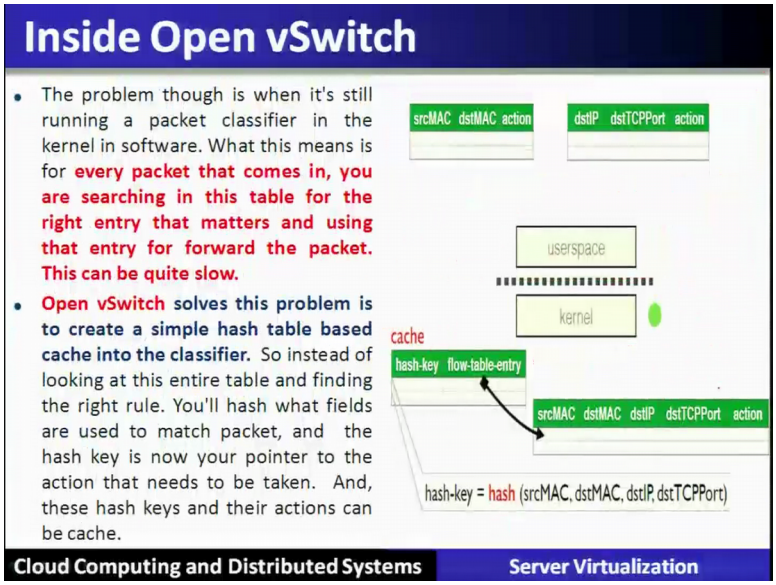
So, let us see the entire division of the task between user space and the kernel space. So, we have only drawn the user space and the kernel space. So, the first packet flow goes to the user space here the several different packet classifiers are consulted. So, some action will be taken based on the MAC address and some maybe depending upon the TCP port address. So, these classifiers will classify the packet based on these type of addresses and the highest priority matching action across this classifier will be used to forward the packets.

So, here all the smart logic in terms of rules and filters are applied here in the different classifier which is residing in basically the user space. Having done this classification once the packet is forwarded this packet will be forwarded that is installed in the kernel. So, this is a simple classification with no priorities the following packets of this flow will never enter the user space seeing only the kernel level classifier. In the kernel level classifier this particular lookup will not be consulted through a big tables, but rather a cache based lookup entry will be will be there based on the hash key.

So, the hashing is basically the constant time search operation or a lookup of the order one can be implemented into the inside the kernel and once it is entered into the table that is in into the cache or a hash table then further accessing will be very fast that is of the order one to those particular packets. So, this way the packet processing into the kernel becomes the cache based hashing scheme which becomes very fast in contrast to

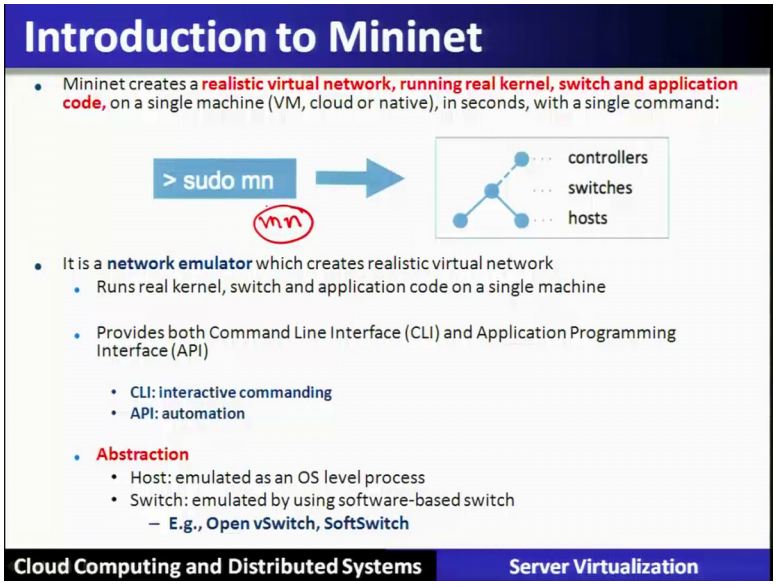
the previous scheme, where the CPU used to consult many tables and the search becomes inefficient and the packet processing takes lot of time.

(Refer Slide Time: 49:11)



So, the problem tough is when it is still running the packet classifier in the kernel, what this means is that for every packet that comes in you are searching the table for the entry that matters and using that particular entry for forwarding the packet. So, that has gone now, so in open vSwitch is to create a cache based simple hash table classifier, so that means here the entire table is not consulted.

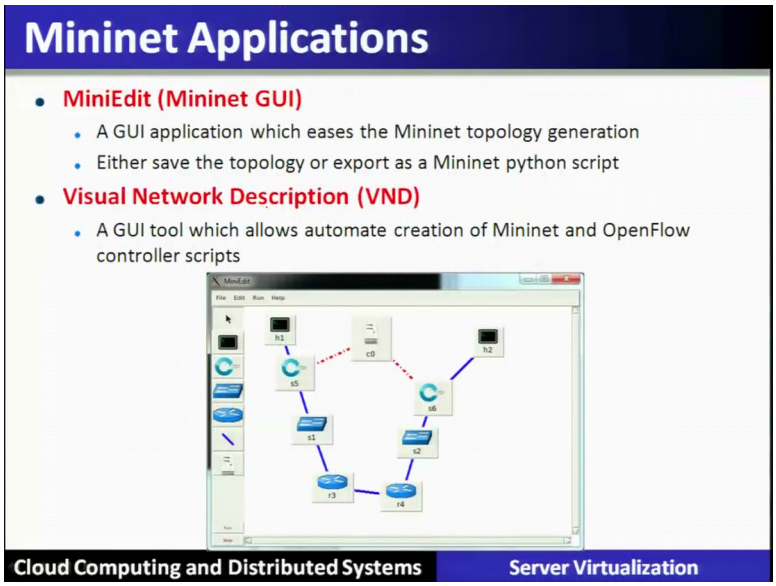
(Refer Slide Time: 49:50)



Now, the next thing which we are now going to discuss is a Mininet. So, Mininet is a emulator for the virtual network, so Mininet creates a realistic virtual network running kernel switch and application code on a single machine and that too in a very efficient manner. So, that is why Mininet is used for experimentation and for the learning also, so if you are interested to implement these concepts that is virtual machine networking or a networking of a virtual machine using emulation then mininet is a useful tool.

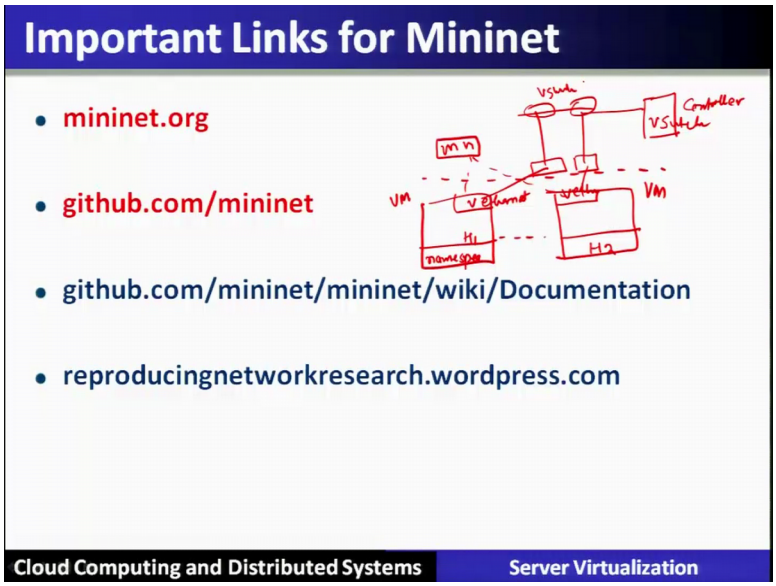
So, Mininet is a network emulator which creates a realistic virtual network. So, you can create your own network virtual network topology using a simple command which is called mn. So, when you run an mn it will create the network emulator for the virtual network. Now this mininet provides you the command line interface and application programming APIs using that you can create your own virtual network topologies. So, here the host is emulated as the operating system level process and also you can implement various switches, which is emulated using software based switch that is you can also implement open vSwitch and soft switches.

(Refer Slide Time: 51:57)



So, this is the visual description you can see using tool you can create your own topology in Mininet.

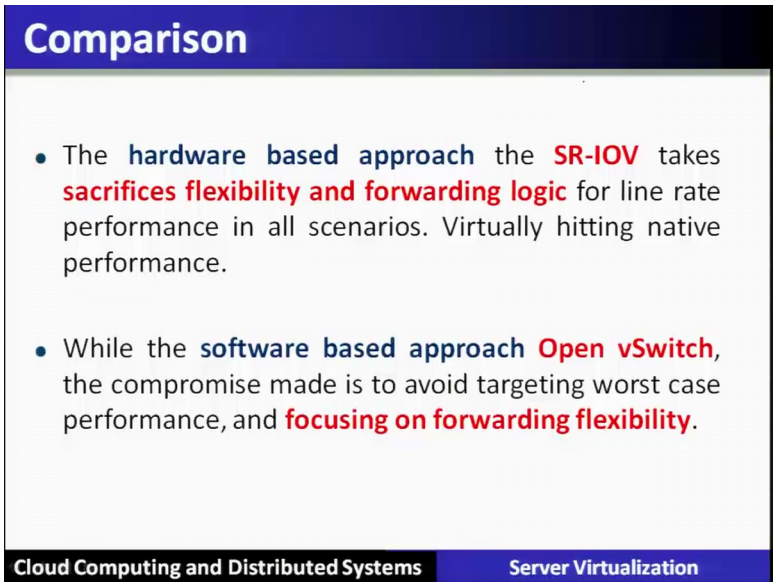
(Refer Slide Time: 52:08)



So, Mininet provides you the way you can create virtual networks you can create hosts, you can attach let us see through the diagram. So, the moment you say Mininet you launch you can create the virtual machines with the namespace, let us say this is H1 and you can create another virtual machine with let us say the name H2.

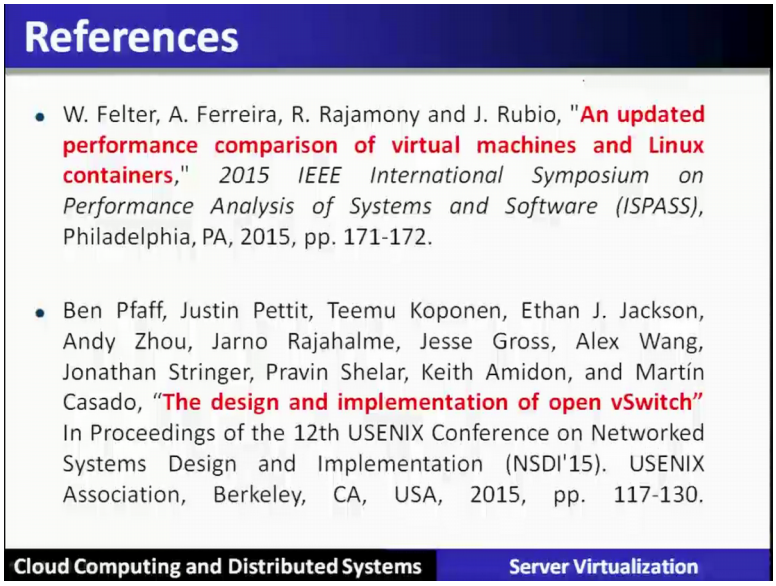
This will be having the ethernet virtual ethernet port and here there will be a physical ethernet which you can connect in the Mininet and intern you can connect to the scenario for control purpose and you can create a vSwitch which can basically create the network topology and the packet processing will be done through those vSwitch. So, all this features are provided in Mininet, so these are the links you can see the details and you can do the experimentation whatever we are discussing here in the server virtualization.

(Refer Slide Time: 54:11)



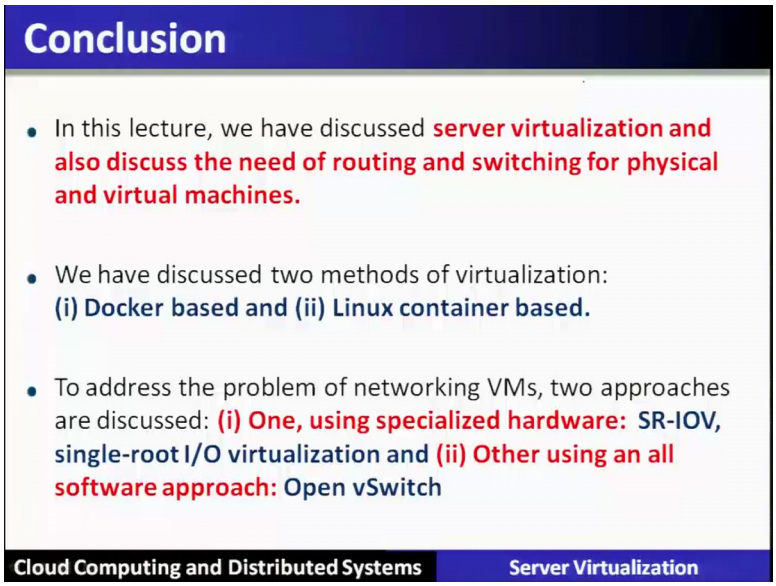
Now, if we compare these 2 schemes hardware based and software based networking schemes which we have seen SR IOV and open vSwitch. So, SR IOV is giving a line rate performance, which is nothing but it is touching up the native performance of a packet forwarding, but it sacrifices the flexibility and also the forwarding logic where as the software based approach that is called vSwitch gives the flexibility and also ensures the fast packet forwarding.

(Refer Slide Time: 55:05)



These are the references which we are used in this part of the lecture conclusion.

(Refer Slide Time: 55:14)



We have covered the server virtualization and we have also covered how the networking of the virtual machines are done, we have also covered 2 different type of virtualization that is Docker based and Linux container based and for that we have also covered the networking virtualization that is through SR IOV and open vSwitch.

Thank you.