Cloud e Datacenter Networking

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI
Laurea Magistrale in Ingegneria Informatica

Prof. Roberto Canonico

Tecnologie di server virtualization e loro impiego in un datacenter

Argomenti della lezione

- Tecnologie di virtualizzazione hypervisor-based
- Tecniche di networking virtuale per VM
- Software virtual switch per sistemi Linux
- Macvlan
- Linux Bridge
- Open vSwitch
- Supporto hardware al virtual networking nelle NIC moderne: SR-IOV

Virtualizzazione

- Per virtualizzazione si intende la creazione di una versione virtuale di una risorsa normalmente fornita fisicamente
- «A framework or methodology of dividing the resources of a computer hardware into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time sharing, partial or complete machine simulation, emulation, quality of service, and many others.»
  Fonte: http://www.kernelthread.com/publications/virtualization/

Hardware Virtualization

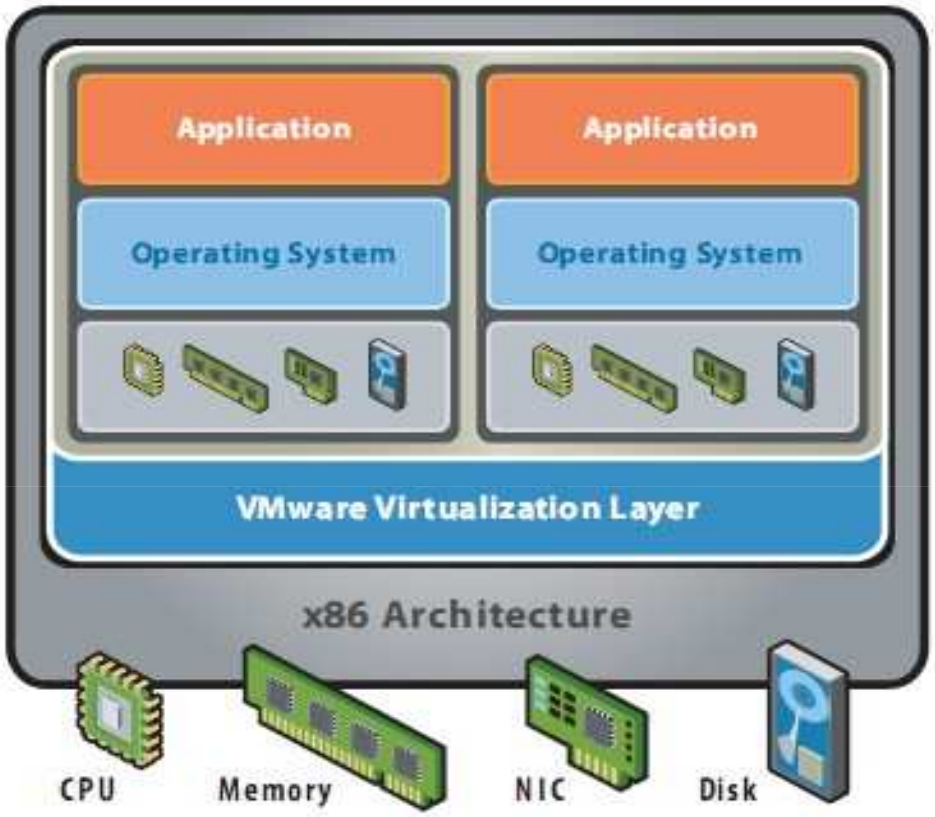
- Quasi qualunque risorsa hardware o software può essere virtualizzata:
  - CPU
  - Memoria RAM
  - Storage su disco rigido
  - Sistema operativo
- Un tipico esempio di virtualizzazione è la divisione di un disco fisso in partizioni logiche
- Per *hardware virtualization* si intendono quelle tecniche che consentono di virtualizzare un'intero computer allo scopo di creare molteplici Macchine Virtuali (*Virtual Machines* o VM) contemporaneamente in esecuzione su uno stesso computer
- Queste tecniche devono riuscire a virtualizzare TUTTE le risorse del computer per creare un hardware «virtuale» con il quale possano interagire le VM

Server Virtualization vs Desktop Virtualization

- Le tecniche di Hardware Virtualization possono essere applicate in due differenti scenari:
  - Server Virtualization
  - Desktop Virtualization
- In entrambi i casi, l'obiettivo è quello di creare delle VM che funzionino come un intero computer
- Nel caso della *Server Virtualization*, dal momento che un server è un computer che primariamente esegue processi server, l'aspetto di virtualizzazione dell'interfaccia grafica è secondario
  - Tipicamente, un computer server non è dotato di un'interfaccia grafica, avvenendo l'interazione esclusivamente attraverso la Command Line Interface a cui si può accedere da remoto attraverso SSH (Secure Shell)
- Nel caso della *Desktop Virtualization*, sono considerati anche la remotizzazione dell'accesso a risorse quali l'interfaccia grafica del sistema operativo (*Desktop*)

Virtual Machines

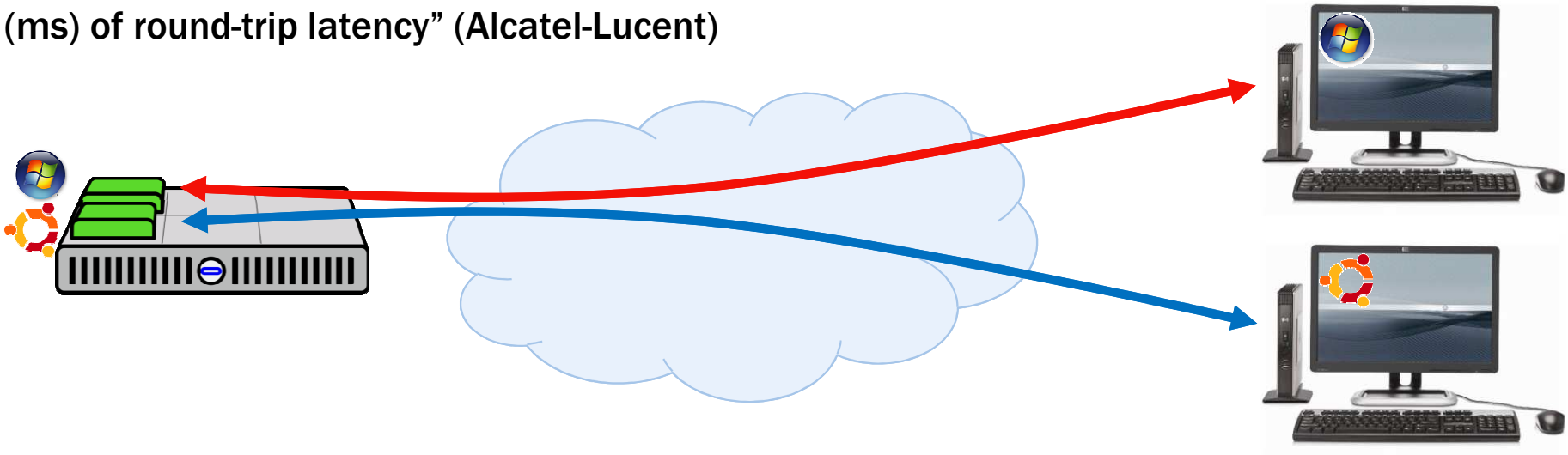
- Un singolo server fisico (host) 'ospita' molte Virtual Machine (VM) (guest)
- Una VM è un contenitore software totalmente isolato che può eseguire i propri sistemi operativi e applicazioni come fosse un computer fisico
- Una VM si comporta esattamente come un computer fisico ed è dotata di proprie CPU, RAM, NIC (Network Interface Card) e dischi virtuali



Fonte dell'immagine:
http://www.vmware.com/pdf/virtualization.pdf

Virtual desktop infrastructure (VDI)

- Lo scenario d'uso detto VDI (*Virtual Desktop Infrastructure*) prevede che un server esegua molteplici VM, ciascuna dotata di un proprio Desktop
- Le VM non sono server ma computer Desktop per uso personale
- L'utente accede alla VM attraverso un computer minimale (*thin client*) costituito da uno schermo grafico, un mouse ed una tastiera, che consente all'utente di accedere al suo desktop in esecuzione su un server remoto attraverso un collegamento di rete
- Il flusso di informazioni verso il thin client può presentare un elevato bit rate
- Affinché l'utente riesca a lavorare senza difficoltà è necessaria una ridotta latenza
  - "A VDI service that offers an experience that is almost indistinguishable from a desktop experience for office suite applications requires 6 Mb/s or more of bandwidth and less than 20 milliseconds (ms) of round-trip latency" (Alcatel-Lucent)



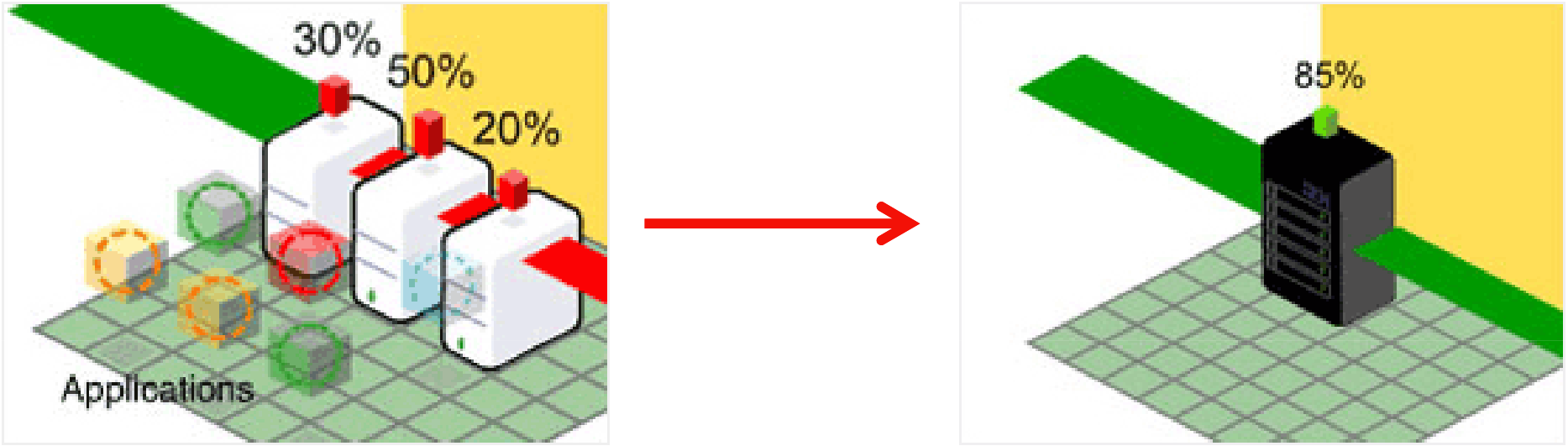
Server virtualization: vantaggi

- Riduzione dei costi:
  - Acquisto (CAPEX)
  - Consumi (OPEX)
    - Elettricità
    - Condizionamento
    - Volume – spazio rack
  - Manutenzione (OPEX)
    - Installazione
    - Configurazione
    - Replica
    - Backup

} Più veloci e quindi meno costosi

- Aumento della disponibilità
  - Riduzione dei tempi di downtime
  - Business Continuity
  - Disaster Recovery
- Maggiore velocità nella messa in esercizio di nuovi servizi

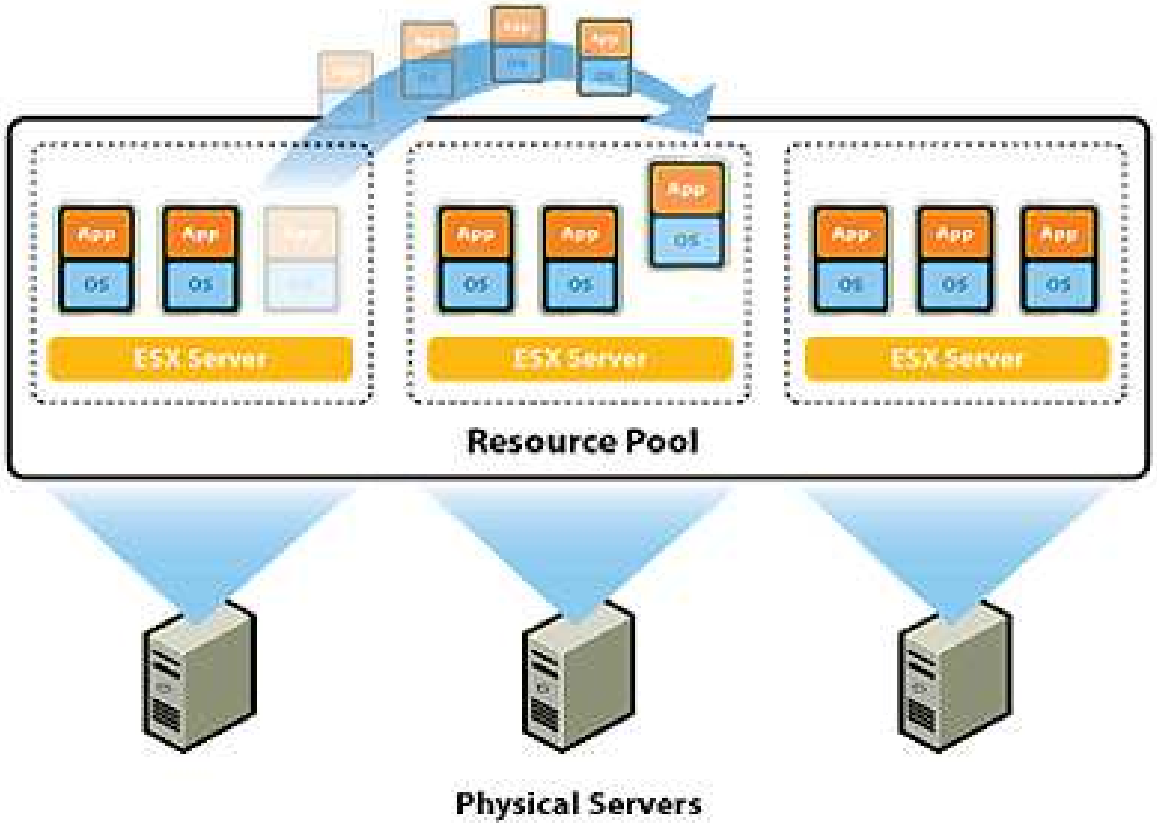
Server virtualization: consolidation



Server consolidation: anziché avere tanti server distinti, ciascuno utilizzato solo per una piccola frazione della sua capacità, si aumenta il livello di utilizzo delle risorse attivando diverse macchine virtuali su un numero ridotto di server fisici, riducendo i consumi energetici e di spazio

Server virtualization: gestione dinamica del carico

- **Live Migration**: spostamento di VM da un server fisico ad un altro senza interruzione del servizio
- Consente di massimizzare, quando possibile, il consolidamento e ridurre i consumi
- Consente l'esecuzione di operazioni di manutenzione hardware ai server senza interrompere i servizi



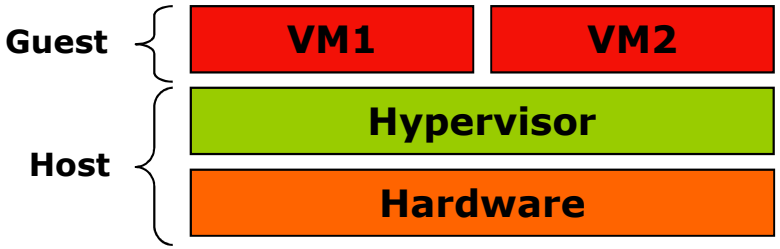
Caratteristiche richieste ad una tecnologia di virtualizzazione

- Isolamento
  - Una VM non deve influenzare in alcun modo le altre VM co-locate (cioè ospitate dallo stesso server fisico)
  - La tecnologia di virtualizzazione deve gestire I problemi che nascono dall'accesso condiviso alle risorse (es. evitare che una VM con l'uso di CPU al 100% produca un rallentamento delle altre VM)
- Efficienza
  - Possibilità di controllare la quantità di risorse utilizzabili da ogni singola VM
- Flessibilità di gestione
  - Possibilità di accendere/spegnere VM in qualunque momento
- Rispetto dei parametri di qualità fissati in uno SLA

Two types of hypervisors

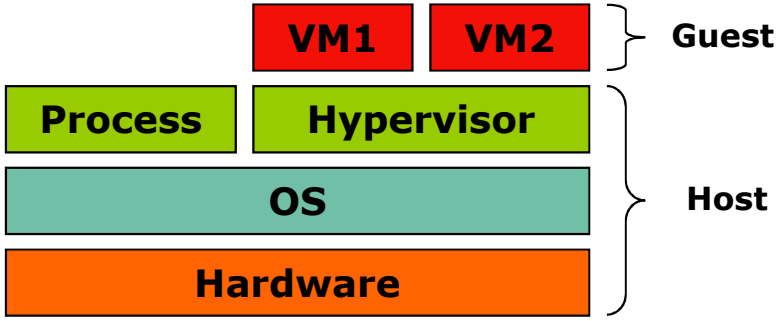
- A **Hypervisor** (or **VMM** – Virtual Machine Monitor) is a software layer that allows several **virtual machines** to run on a **physical machine**
  - The physical OS and hardware are called the **Host**
  - The virtual machine OS and applications are called the **Guest**
- Two types of hypervisors are commonly recognized: Type-1 and Type-2

Type 1 (bare-metal)



VMware ESX, Microsoft Hyper-V, Xen Server

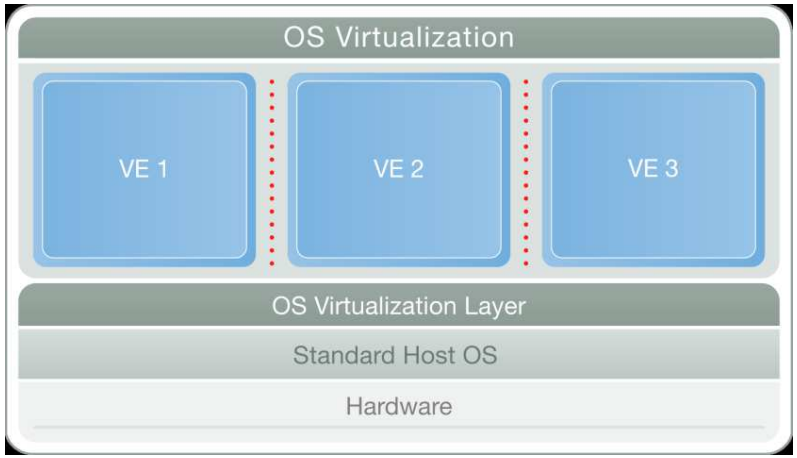
Type 2 (hosted)



VMware Workstation, Microsoft Virtual PC,
Sun VirtualBox, QEMU, KVM

A different approach to virtualization: containers

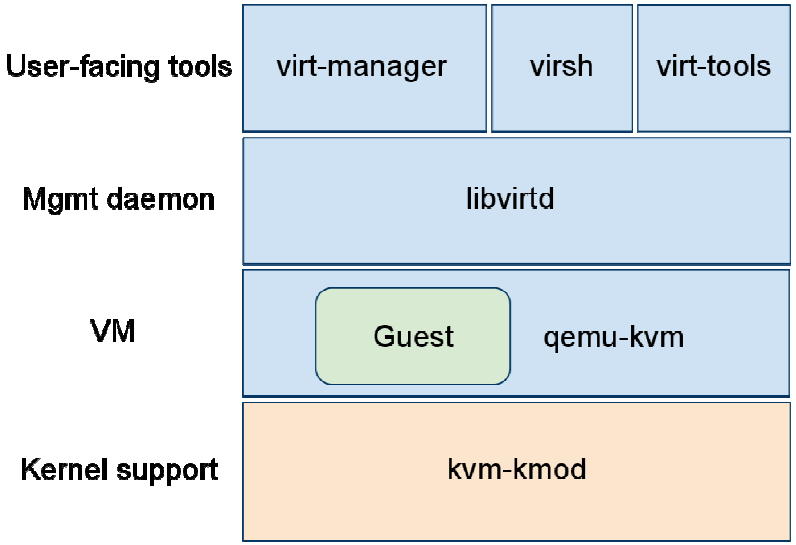
- Lightweight virtualization provided by the OS
- **One** real HW (no virtual HW), **one** kernel, **many** userspace instances
- Less overhead, best performance, more concurrent virtualized execution environments on the same hardware



- An idea that has been implemented in several ways
  - FreeBSD jails
  - Linux-Vserver
  - OpenVZ / Parallels Containers
  - Solaris Containers/Zones
  - IBM AIX6 WPARs

KVM: Kernel-based Virtual Machine for Linux

- KVM is a full virtualization solution that turns a Linux kernel into a hypervisor using a kernel module (kvm.ko)
- Open source project: http://www.linux-kvm.org
- Implemented in Linux Kernel since 2.6.20
- The introduction of the KVM is an interesting evolution of Linux, as it represents the first virtualization technology that is part of the mainline Linux kernel
- Two components: a kernel module (kvm.ko) and a user process (QEMU)
- In practice, other user-level tools are needed (libvirtd, virsh, virt-tools, ...)
- When run on CPUs that supports virtualization, Linux and Windows guests are supported

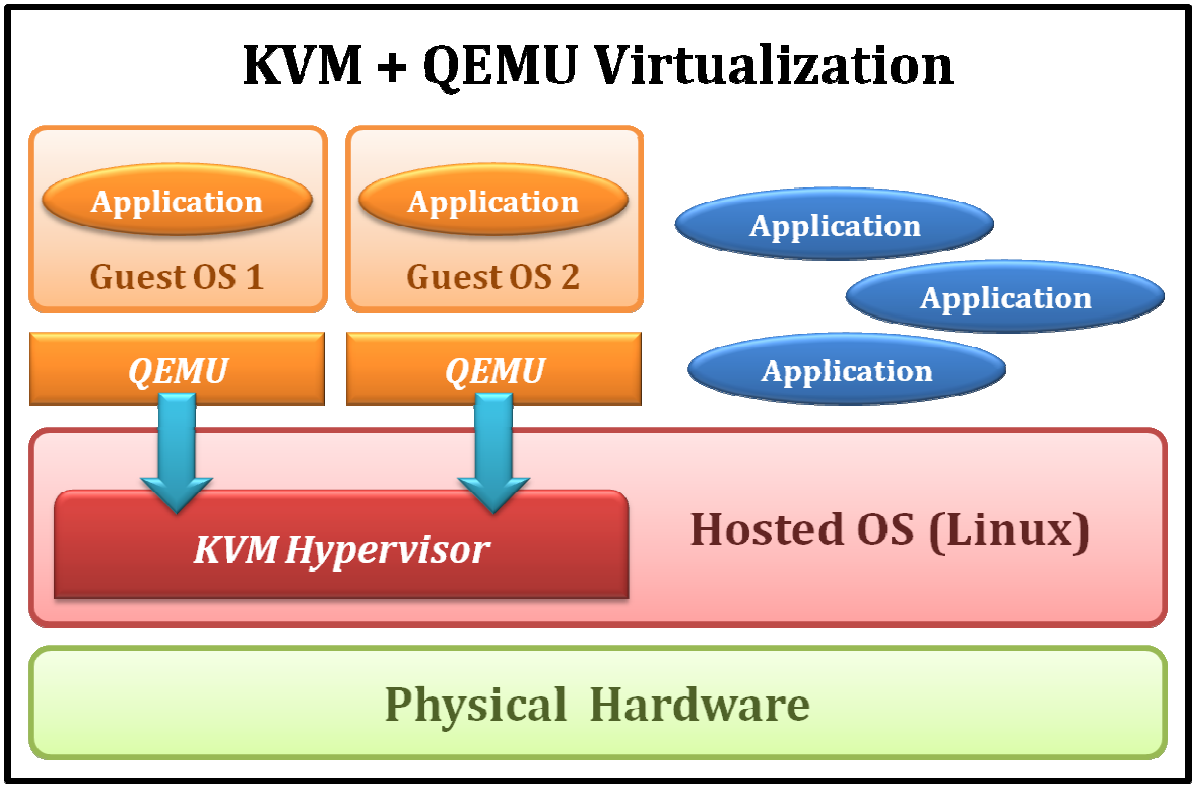


KVM kernel modules (1)

- KVM is based on a loadable kernel module (kvm.ko) that allows other guest operating systems to run in user-space
- kvm.ko exposes virtualized hardware through the /dev/kvm character device
- The KVM module introduces a third execution mode into the kernel
  - Where vanilla Linux kernels support *kernel mode* and *user mode*, KVM introduces a *guest mode*
- The guest mode is used to execute all non-I/O guest code, where normal user mode supports I/O for guests
- Zero impact on host kernel
- Guests are scheduled as regular processes
  - kill(1), top(1) work as expected
- The guest operating system interfaces to the KVM module using a modified QEMU process for PC hardware emulation

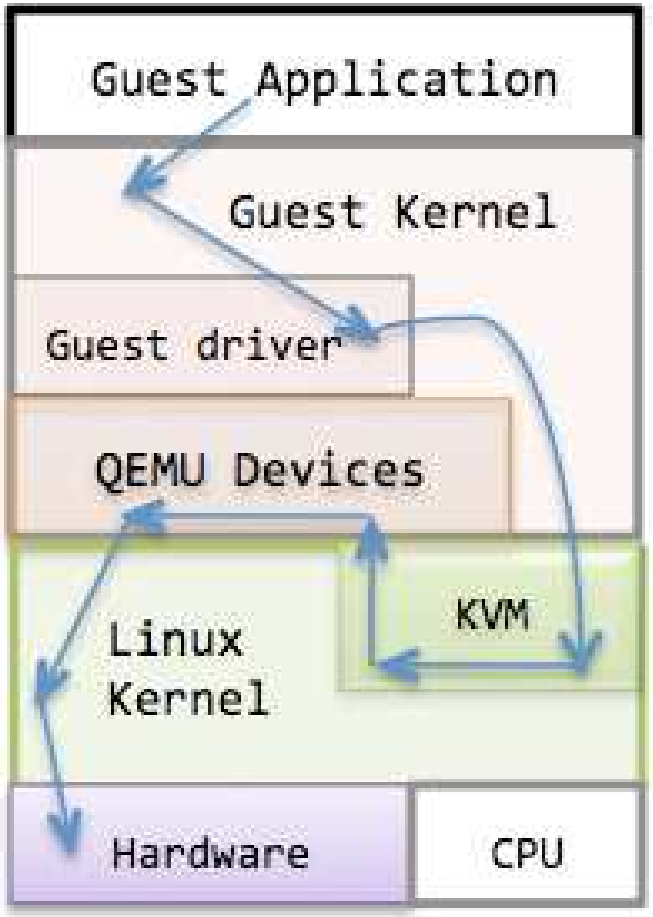
KVM kernel modules (2)

- kvm.ko
  - provides the core virtualization infrastructure
- kvm-intel.ko / kvm-amd.ko
  - processor specific modules



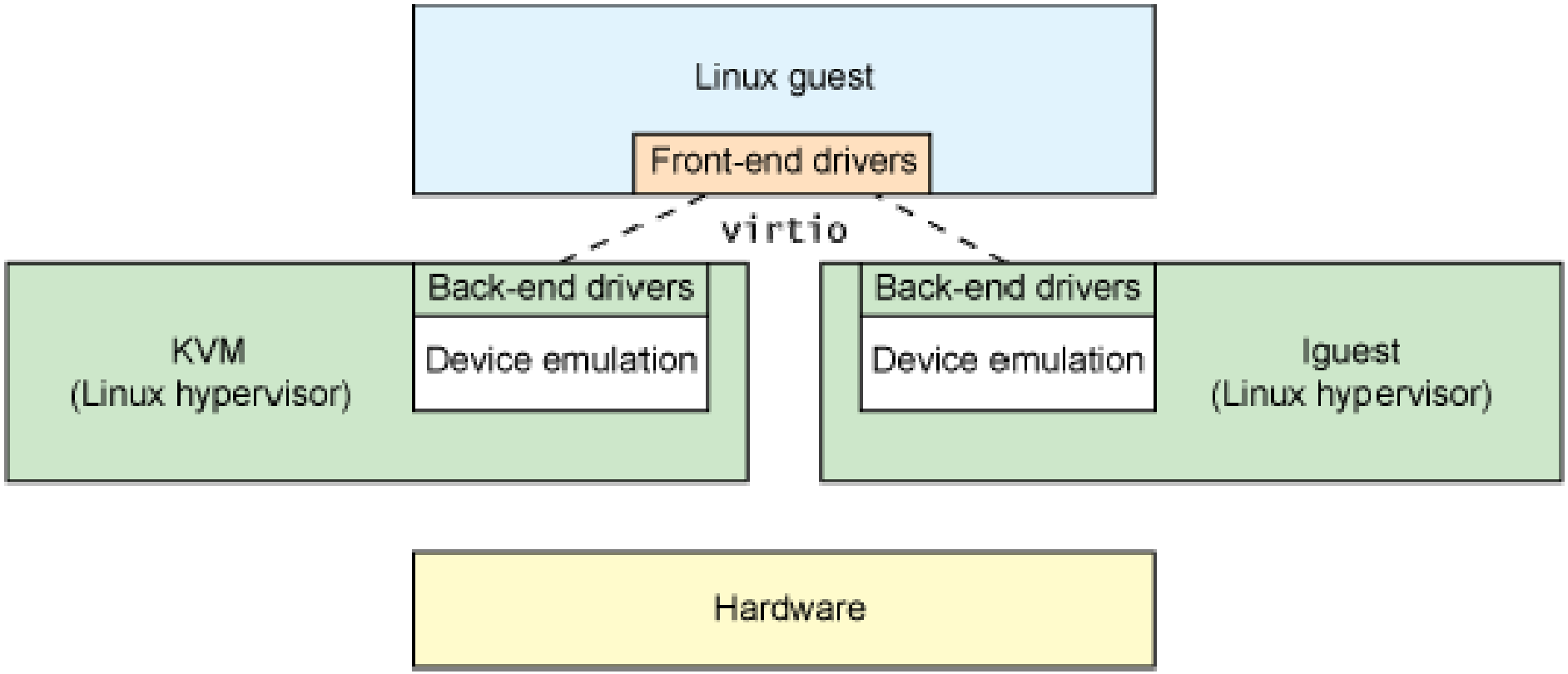
I/O in KVM (full virtualization)

- Original approach with full-virtualization
  - Guest hardware accesses are intercepted by KVM
  - QEMU emulates hardware behavior of common devices
    - RTL 8139
    - PIIX4 IDE
    - Cirrus Logic VGA



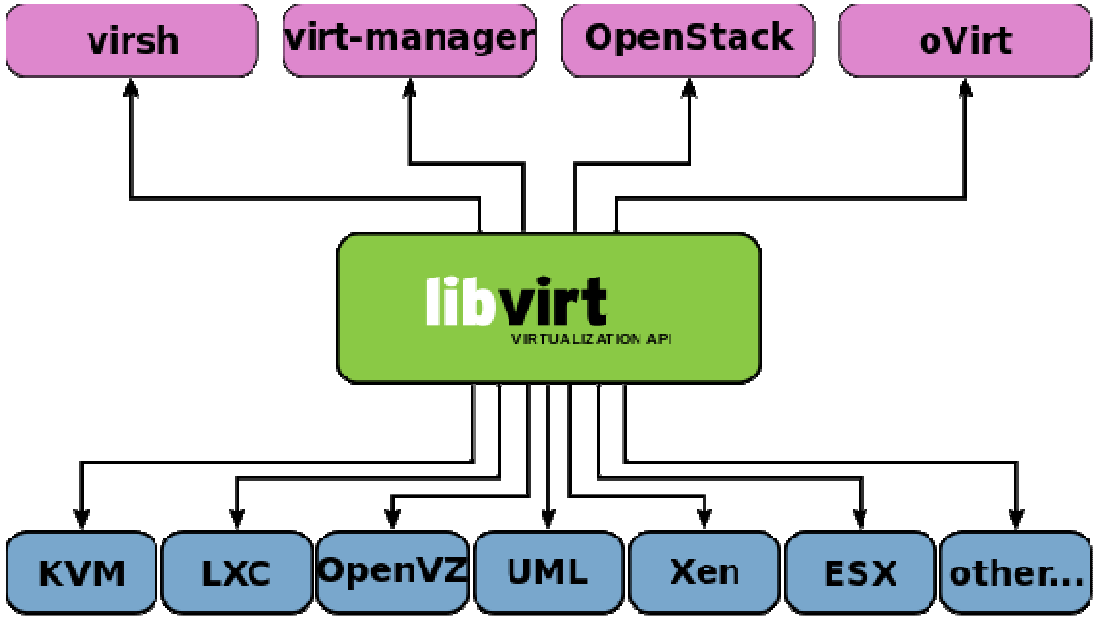
I/O in KVM (virtio)

- New approach with para-virtualization



Libvirt – API for managing VMs

- Libvirt is an open source API, daemon and management tool for managing platform virtualization
- Implemented as a C library, can be used b programs written in many languages, such as Python, Perl, OCaml, Ruby, Java, and PHP
- Widely used in the orchestration layer of hypervisors in the development of a cloud-based solution (e.g. in OpenStack)
- Can be used to manage KVM, Xen, VMware ESX, QEMU and other virtualization technologies
- Two User Interfaces:
  - Graphical Interface: **virt-manager**
  - Command line interface: **virsh**

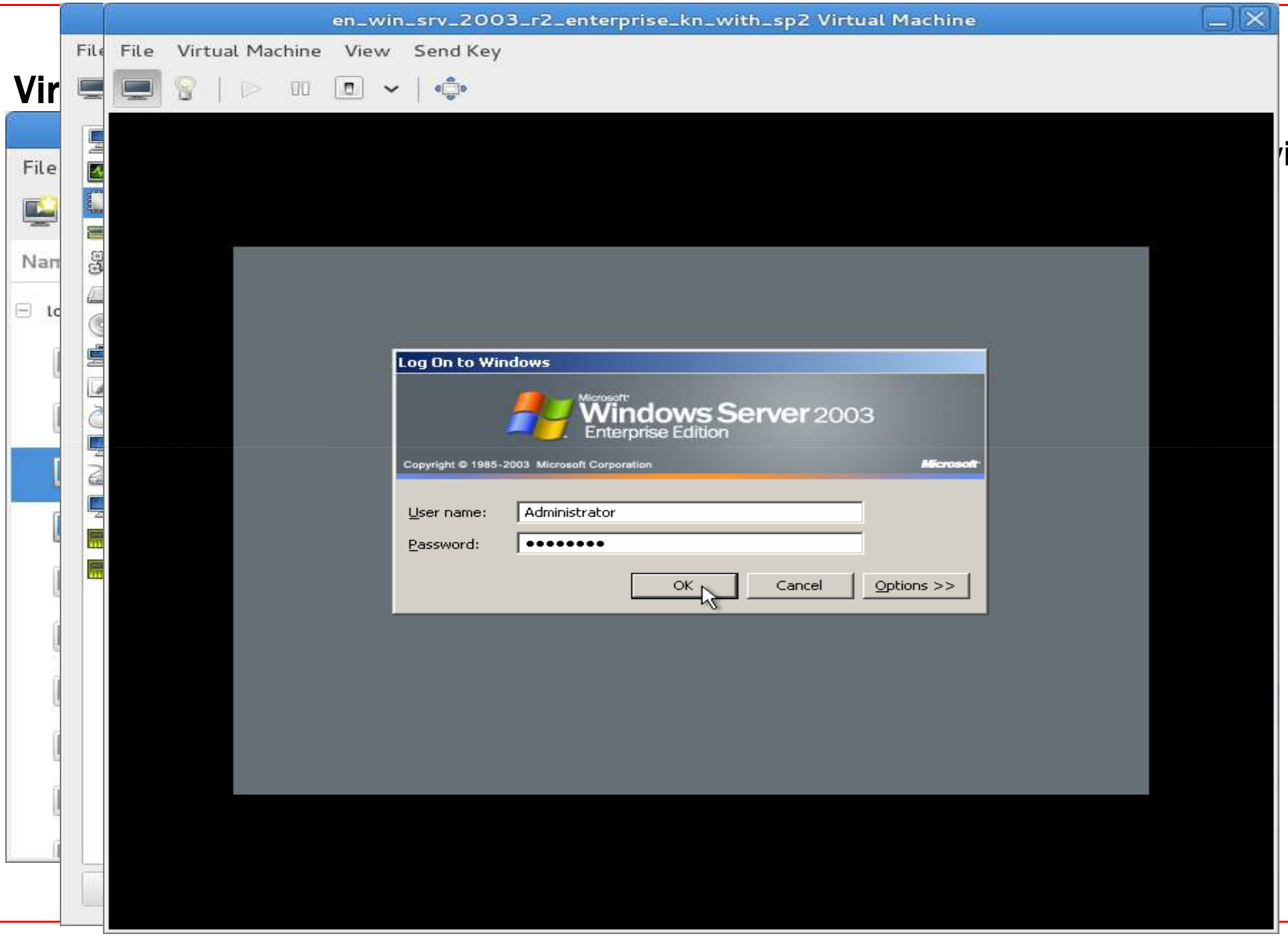


virt-manager

- **Virtual Machine Manager**
  - is a desktop-driven virtual machine manager with which users can manage virtual machines (VMs)
- **Functions**
  - create, edit, start and stop VMs
- **virt-manager's supporting tools**
  - **virt-install** tool
  - **virt-clone** tool
  - **virt-viewer** application

virt-manager

- Vir
- virtual

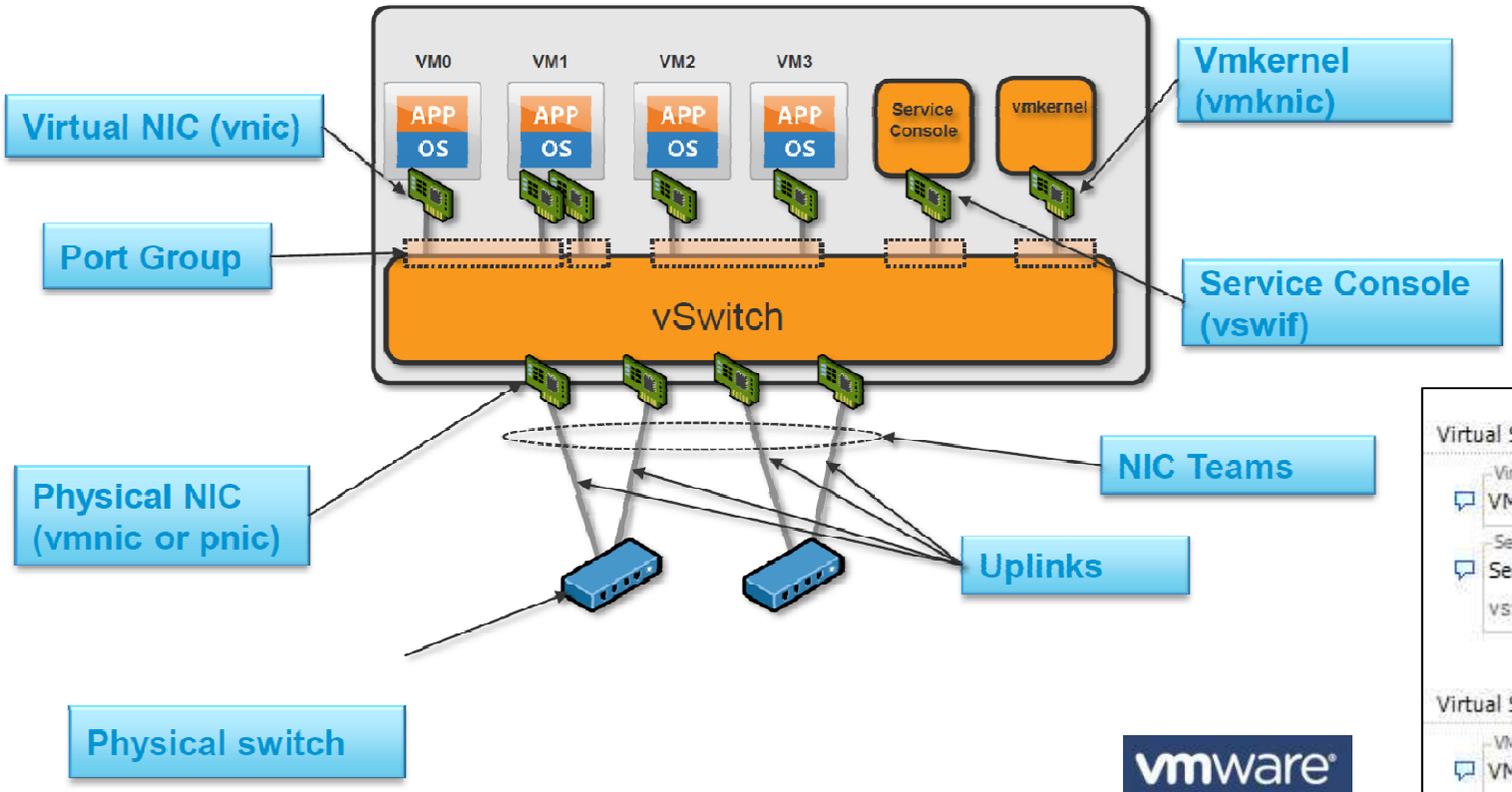


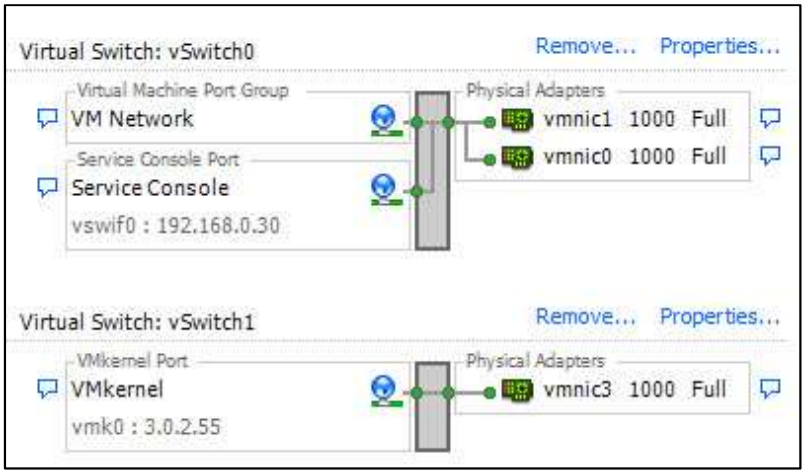
virsh

- virsh is a command-line tool used to manage domains (i.e. VMs)
- virsh commands need root privileges to be executed
- Common used commands:
  - `virsh create`
    - start a VM from an XML descriptor file
  - `virsh destroy`
  - `virsh list`
    - list all the running VMs
  - `virsh console`
    - connect to a VM
  - `virsh reboot`
  - `virsh shutdown`
  - …

Hypervisors and VM networking (1)

- In a physical host with several VMs, each VM has it own virtual NIC(s) (or **vNIC**s)
- Virtual NICs are connected to the host physical NIC(s) by means of a virtual switch (or **vSwitch**) whose job is to dispatch packets from/to VMs according to their virtual MAC addresses
- A single hypervisor may be configured with several vSwitches





Hypervisors and VM networking (2)

- A hypervisor creates virtualized network devices: vNICs and vSwitches
- Many VMs connected by virtual network devices create a *virtual network*
- Virtual network interface card (vNIC)
  - Hypervisor can create one or more vNICs for each VM
  - The vNIC provides the networking capabilities of the VM
  - Each vNIC is identical to a physical NIC
- Virtual switch(vSwitch)
  - Switches also can be virtualized as a virtual switch
  - Each vNIC is connected to a vSwitch port
  - A vSwitch may be connected to an external physical network through a physical NIC (pNIC) of the hypervisor

VMware terminology

- Virtual Network Adapters
  - vNic – VM's interface to the network
  - vmknic – vSphere hypervisor's interface to network (nfs, iSCSI, vMotion, FT, Management)
  - vswif – Interface for Service Console (not present on ESXi)
- Physical Network Adapter
  - pNic – for communicating with entities outside ESX/ESXi host
- Virtual Switch
  - vSwitch – forwards packets between vNics, vmknics, and pNics
- Port Group
  - Group of ports sharing the same configuration (e.g. vlan)
- Uplinks: connections to physical switches
- NIC Team: a group of pNics connected to the same physical network

VMware: 3 types of Virtual Switches

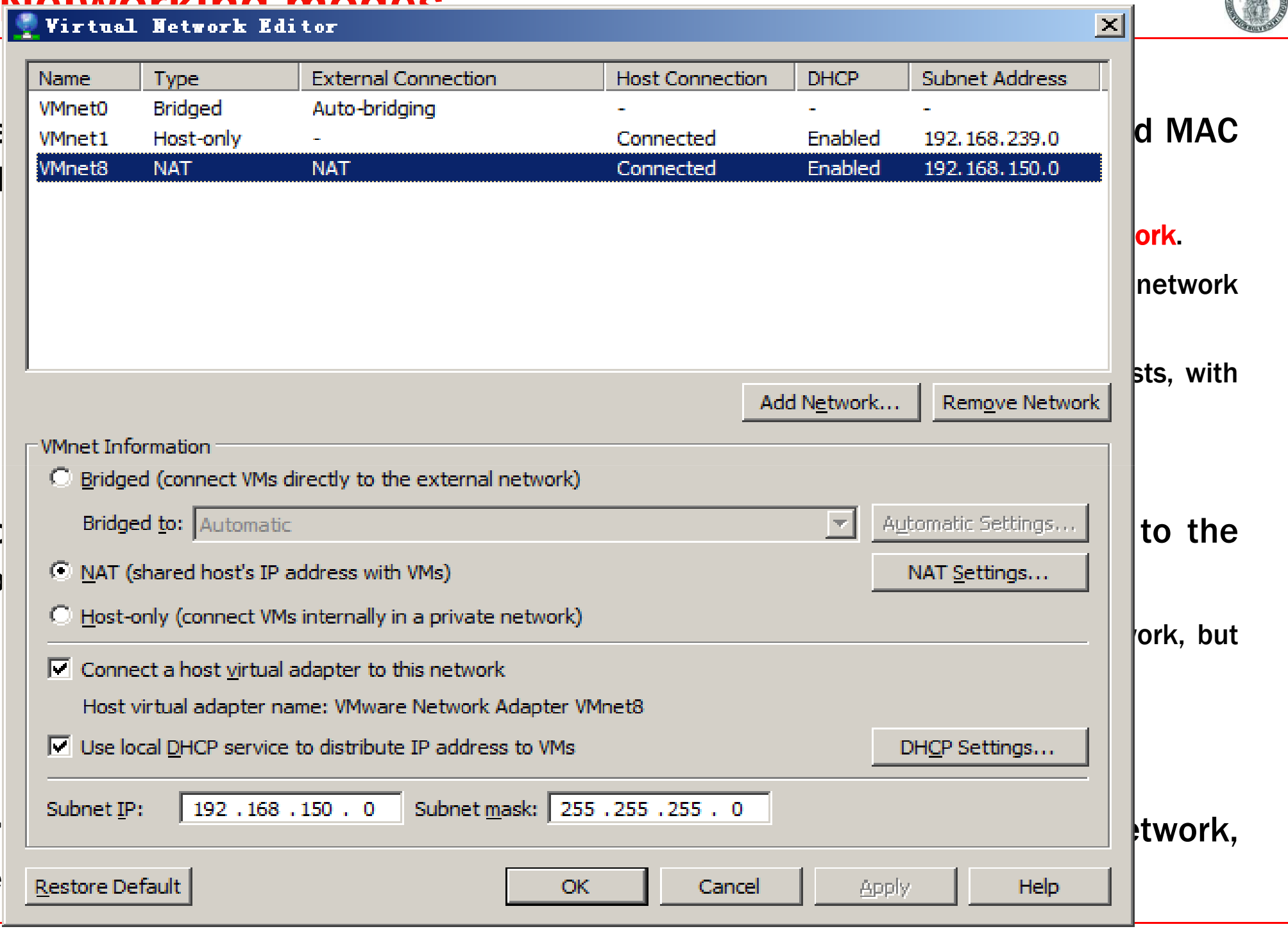
- vNetwork Standard Switch (vSS)
  - Created and managed on a per-host basis
  - Support basic features such as VLAN, NIC teaming, port security
- vNetwork Distributed Switch (vDS)
  - Created and managed at vSphere vCenter
  - Supports all vSS features and more (PVLAN, traffic management, etc.)
  - NOTE: vSS/vDS share same etherswitch module, only control path differ
- Cisco Nexus 1000v (N1K)
  - Created and managed by VSM (either VM or hardware/Nexus 1010)
  - Supports features typically available in Cisco hardware switches

VM networking modes

- There are 3 popular methods to connect a VM to the host and to the external networks to which the host is connected
- **Bridged**: under the Bridged method, the VM will directly contact the DHCP server of the external physical network and apply for a unique local IP address in the external network; the VM will be then able to directly access the external network; this is the preferred connection method, if we run any server in the VM
- **NAT (Network Address Translation)**: under this method, the VM accesses the host's external network with the IP address of the host; within the host, we have a virtual private network involving the host and the VMs running on it. The other hosts in the external network cannot directly access the VM and they have to go through the NAT process at the host; in other words, the host PC acts as the first-stop gateway router for the VM
- **Host-only**: This method is same as the NAT except the VMs cannot access the external network as the host does not act as a NAT router for the VMs; communication is only allowed among VMs running in the same host

VM Networking modes

- Ne … d MAC ad …
- Ho … to the ho …
- Br … twork, se …

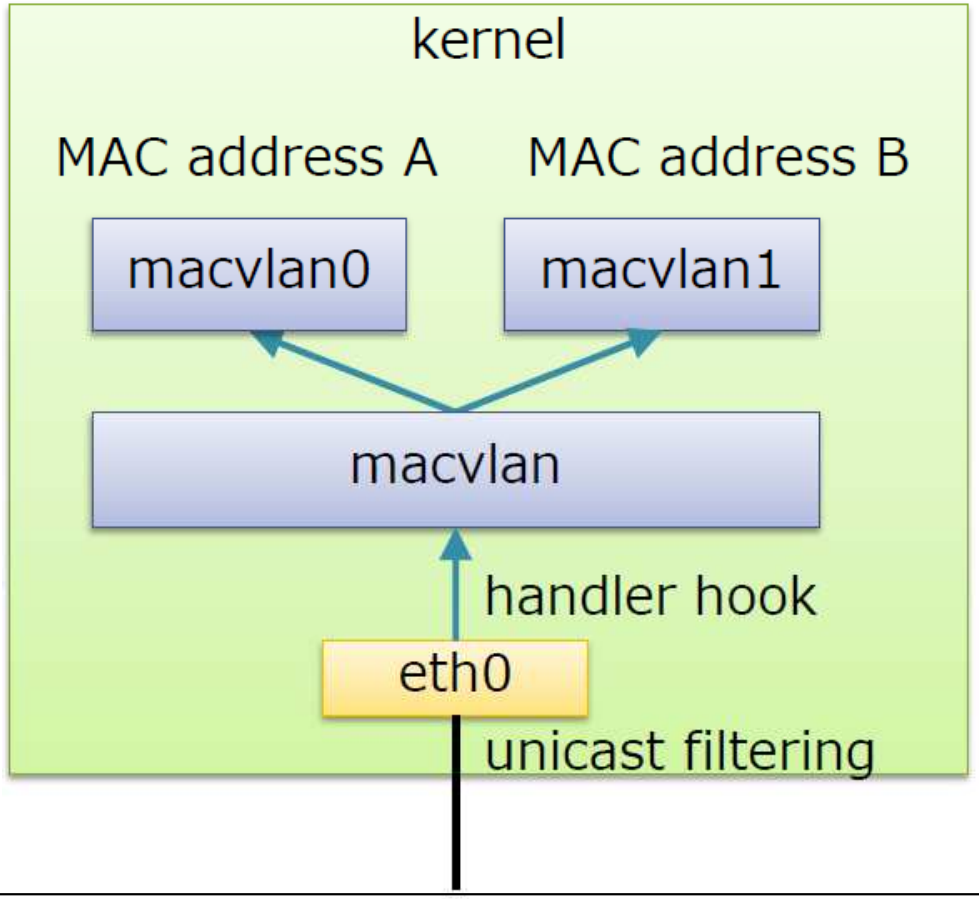


Linux: 3 types of virtual switches

- Macvlan
- Linux Bridge
- Open vSwitch

macvlan

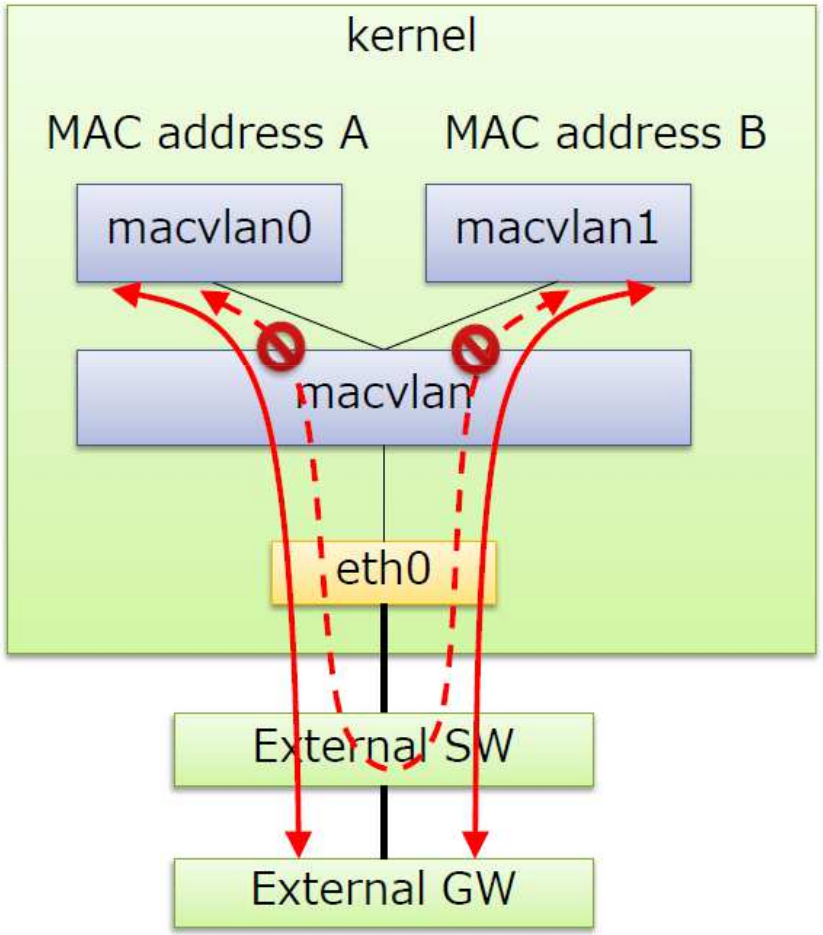
- A mechanisms that creates VLANs in which membership is associated to MAC addresses of VM virtual NICs
- Packets are not tagged (802.1q)
- If the physical NIC allows filtering of multiple unicast MAC addresses, this feature is used instead of promiscuos mode
- Four modes of operation:
  - Private
  - Vepa
  - Bridged
  - Passthru



Toshiaki Makita. *Virtual switching technologies and Linux bridge.* NTT Open Source Software Center.

Macvlan: private mode

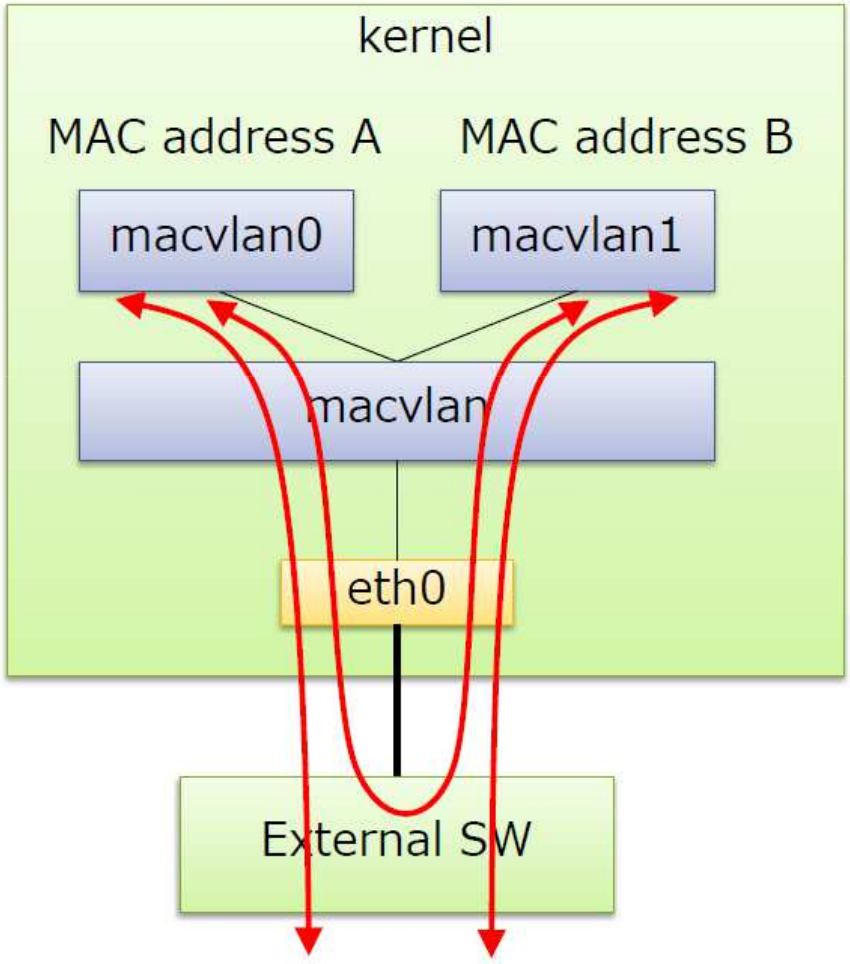
- In private mode the macvlan device does not forward packets among its ports
- VM-to-VM communication needs an external gateway device
  - MAC addresses A and B belong to different VLANs



Toshiaki Makita. *Virtual switching technologies and Linux bridge*. NTT Open Source Software Center.

Macvlan: vepa mode

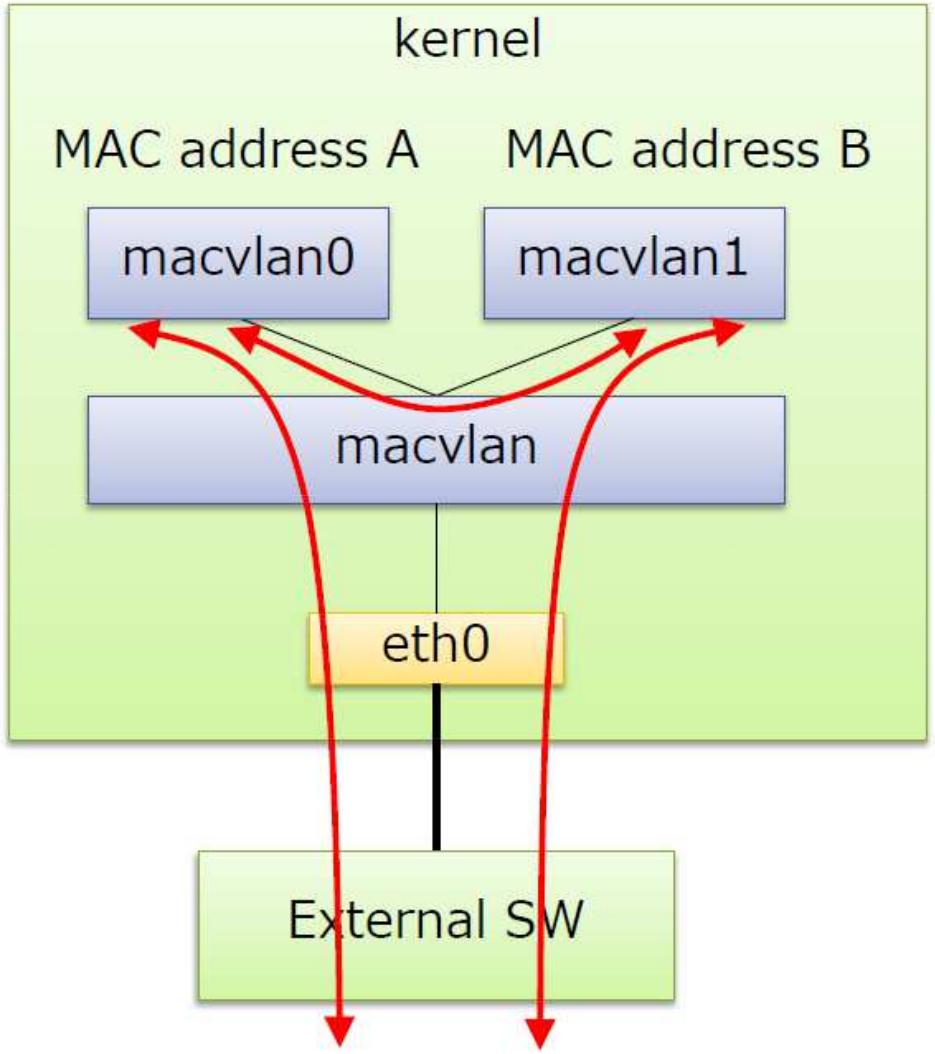
- In vepa mode the macvlan device does not forward packets among its ports
- VM-to-VM communication may happen through an external switch



Toshiaki Makita. *Virtual switching technologies and Linux bridge*. NTT Open Source Software Center.

Macvlan: bridge mode

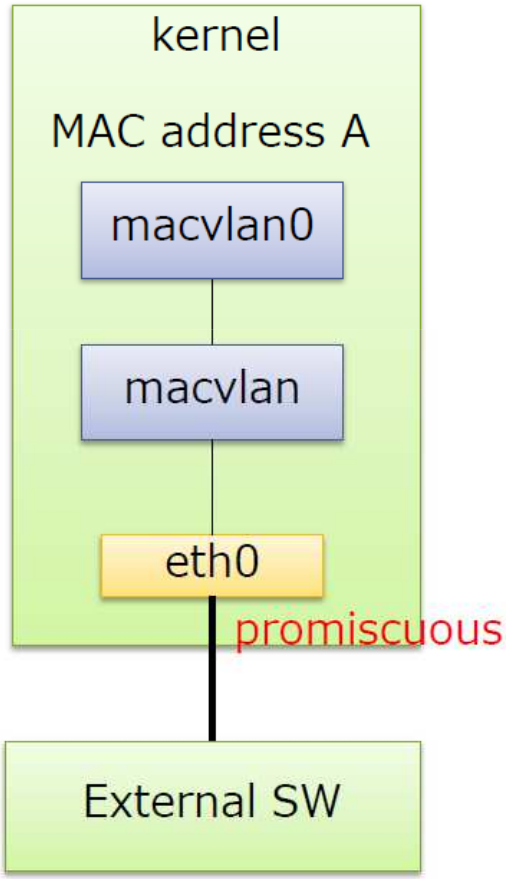
- In bridge mode the macvlan device is able to forward packets among its ports
- VM-to-VM communication may happen internally through macvlan device
- The macvlan device bridging capabilities are minimal:
  - No STP
  - Only one uplink
  - No source MAC learning



Toshiaki Makita. *Virtual switching technologies and Linux bridge*. NTT Open Source Software Center.

Macvlan: passthru mode

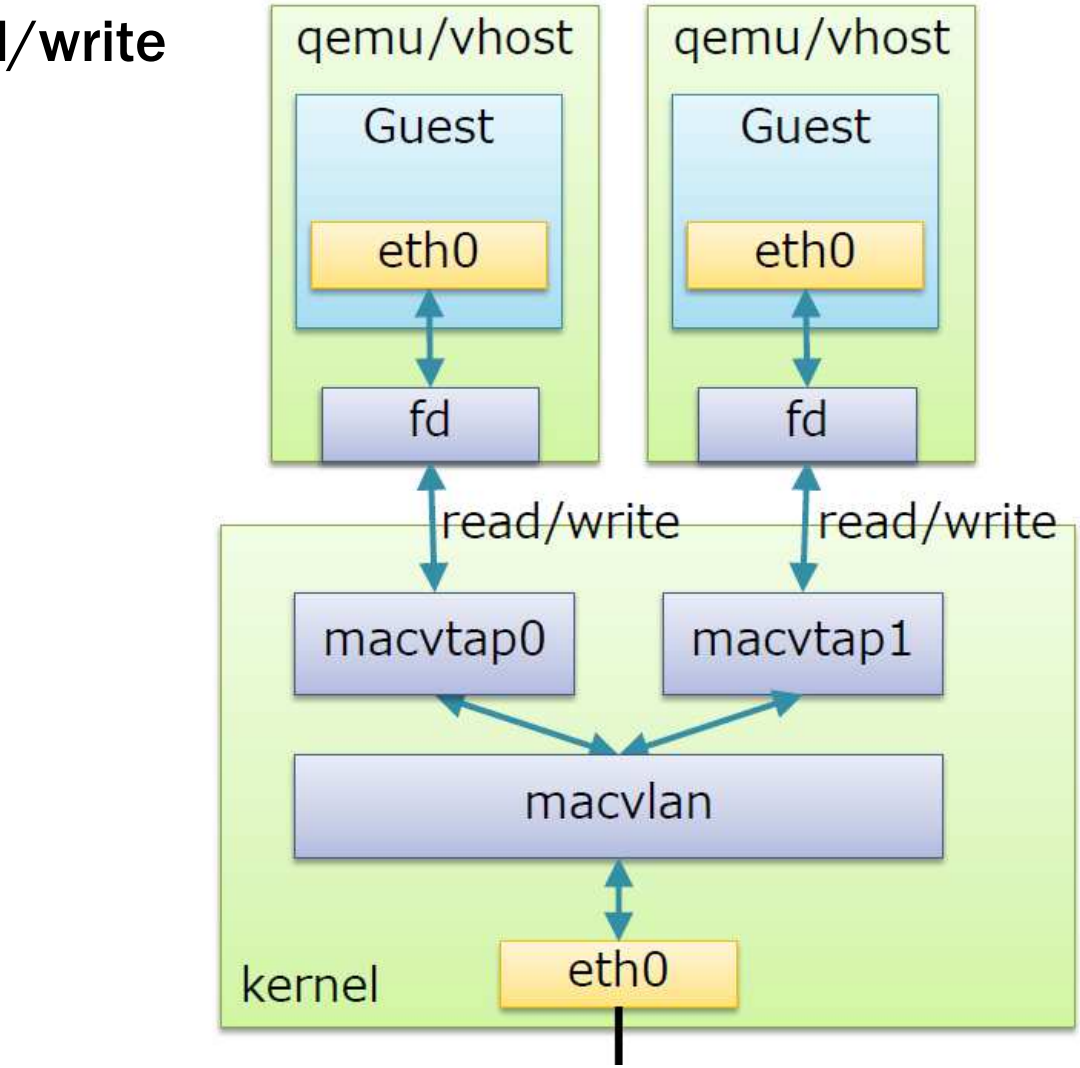
- In passthru mode only one virtual device per macvlan is allowed
- The physical NIC is put in promiscuos mode



Toshiaki Makita. *Virtual switching technologies and Linux bridge*. NTT Open Source Software Center.

KVM and macvlan with TAP interfaces

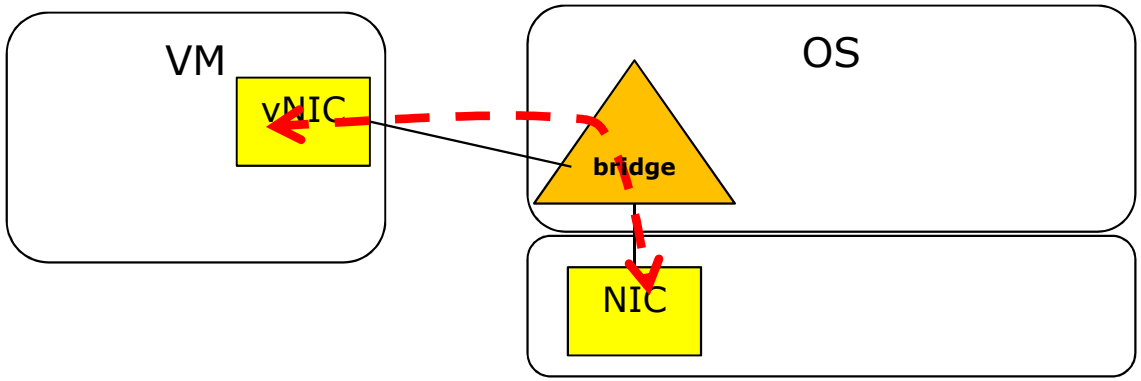
- VMs could be connected to the macvlan device not directly but through virtual TAP devices that expose a file read/write interface
- packet reception → file read
- packet transmission → file write



Toshiaki Makita. *Virtual switching technologies and Linux bridge*. NTT Open Source Software Center.

Linux Bridge

- Linux Bridge
  - is a **virtual network device** working at layer 2
  - works as an Ethernet **physical switch**
- A Linux Bridge can bind other Linux network device as a slave device, and virtualize the slave device as a port
  - using promiscuous mode that allows to receive all packets
- To install Linux Bridge in a Debian-based Linux distribution:
  - `$ sudo apt-get install bridge-utils`
- bridge-utilities is a program that implements a subset of the IEEE 802.1d standard and also comprises STP (*Spanning Tree Protocol*)
- bridge-utilities consists in a Kernel module and a user space application (*brctl*)

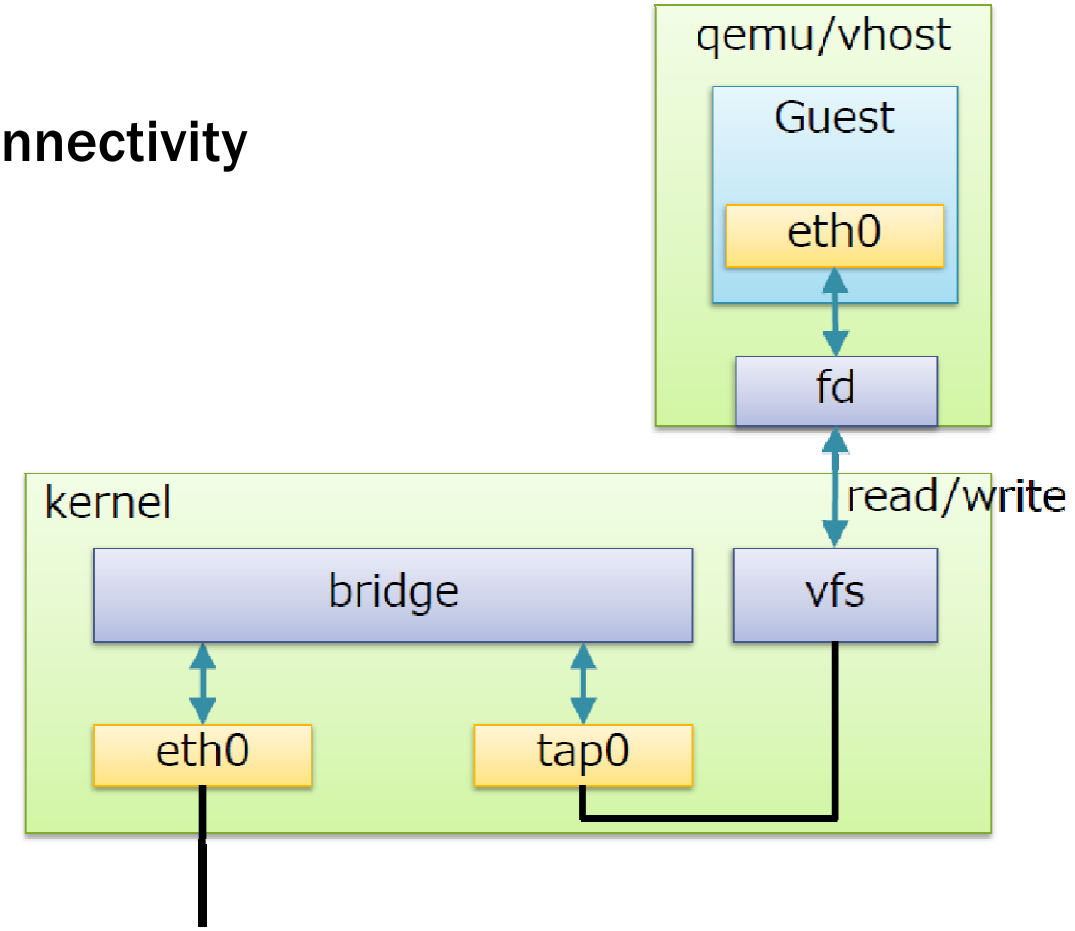


brctl basic commands

- **Create/destroy a bridge device:**
- `$ brctl addbr bridge_name`
- `$ brctl delbr bridge_name`
- **Add/delete interface to a bridge device:**
- `$ brctl addif bridge_name device_name`
- `$ brctl delif bridge_name device_name`
- **Show devices in a bridge:**
- `$ brctl show`
- **Show the forwarding DB:**
- `$ brctl showmacs bridge_name`

KVM and Linux bridge with TAP interfaces

- VMs create an internal connection to a virtual TAP device
- The TAP device is configured as a port for the Linux Bridge
- VM-to-VM communication may happen through the Linux Bridge
- A physical NIC (e.g. *eth0*) provides connectivity towards the rest of the world



Toshiaki Makita. ***Virtual switching technologies and Linux bridge.*** NTT Open Source Software Center.

Tap devices in a KVM hypervisor

- Each VM corresponds to a tap device

```
[root@compute2 ~]# ifconfig |grep tap
tap0920d05a-38: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap1c52b862-e1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap2efedf47-39: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap45787583-0b: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap5140c9f1-87: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap56b1fa1f-e5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap69383260-61: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap6bbe47a7-f2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap7b92e41b-a3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap8213fc6e-f9: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap833640f0-24: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap915224e1-da: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tap9bc4ac99-b9: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tapb8e468e6-59: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tapbb4b88b7-1f: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tapcaf75f1d-9b: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tapdaa7ce9a-7c: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tapdea66e2e-63: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
tape28c2f3a-a5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

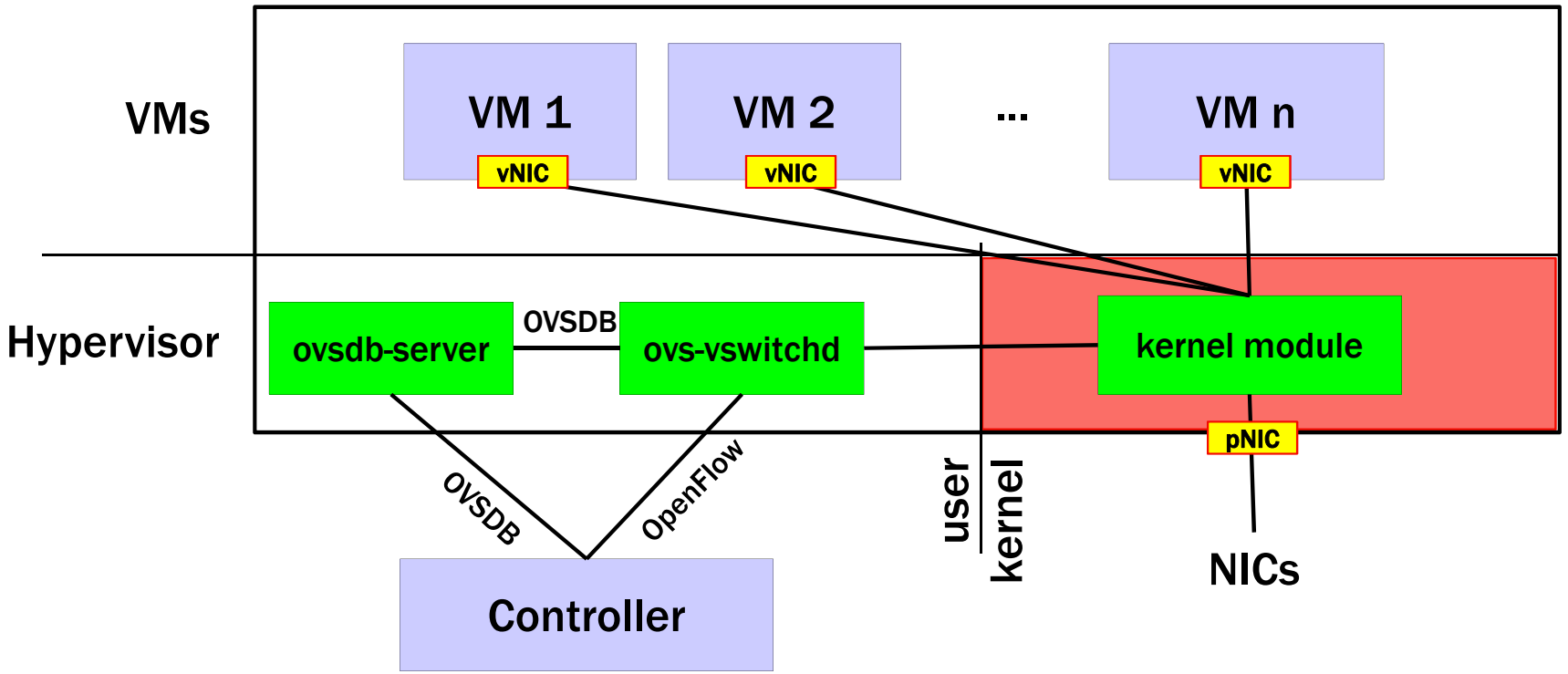
Virtual switches performance and SR-IOV

- Since a virtual switch requires some CPU processing to select the outgoing packet queue and to copy packets from one physical NIC to a virtual NIC (or viceversa), CPU power may limit the aggregate VM throughput
- At 10 Gbps, the time it takes to transmit an 84 bytes packet is 67 ns
- A single access to memory may require in the order of 10 ns
- To improve packet processing throughput in the case of many VMs, a new technology has been developed: SR-IOV (*Single Root I/O Virtualization*)
- SR-IOV relies on modern NICs

Open vSwitch

- “Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (e.g. NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag).”
- Key design decision of Open vSwitch is to partition functions among kernel and user space
  - Performance-limiting operations (*packet forwarding*) executed in kernel space
  - Control place operations are executed in user space
- In kernel space, to speedup forwarding decisions, these are taken by calculating a hash function on the tuple (src-MAC, dst-MAC, dst-IP, dsp-TCP-port) and stored in a cache to keep forwarding decisions within kernel space
  - Only in case of a cache miss (fist packet of a flow), user-space classifiers executed
  - Subsequent packets of a flow match a cached rule

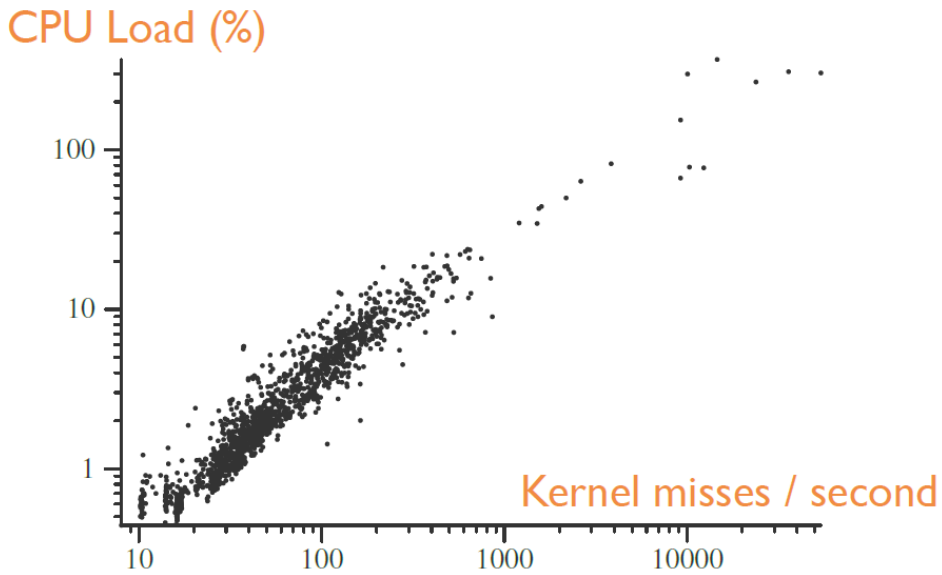
Open vSwitch Architecture



User space classifiers may be programmed and controlled by an external entity (OpenFlow controller)

Open vSwitch performance

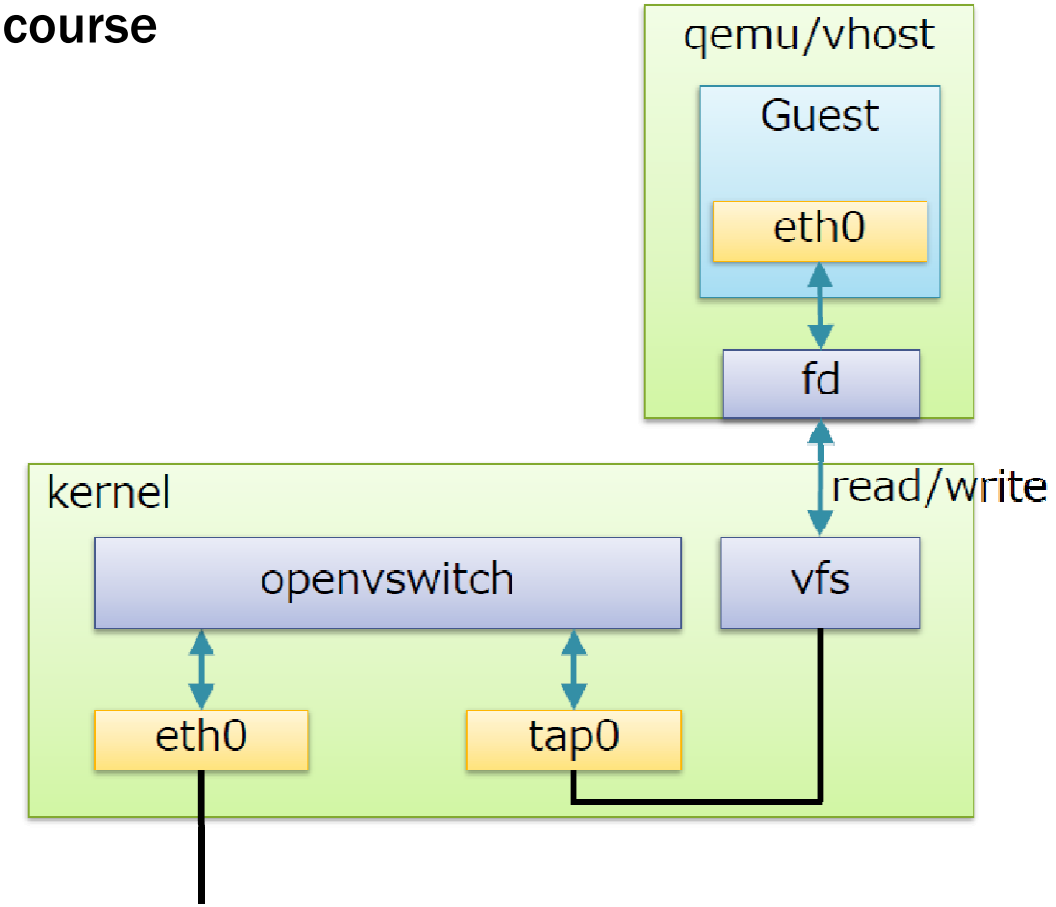
- Performance evaluation of Open vSwitch presented in [*] on a real deployment of a large number (> 1000) of hypervisor nodes dealing with huge traffic (24h)
- The more the caching function within the kernel is bypassed (cache miss), the more the host CPU is loaded
- However, for the vast majority of cases, CPU is never loaded more than 20%
- Over 80% of nodes with a CPU load less than 5%



[*] [The Design and Implementation of Open vSwitch. Pfaff et. al, USENIX NSDI 2015]

KVM and openvswitch with TAP interface

- Same configuration as for the Linux Bridge
- Plug-and-play replacement of Linux Bridge with greater flexibility
  - See OpenFlow later on in the course



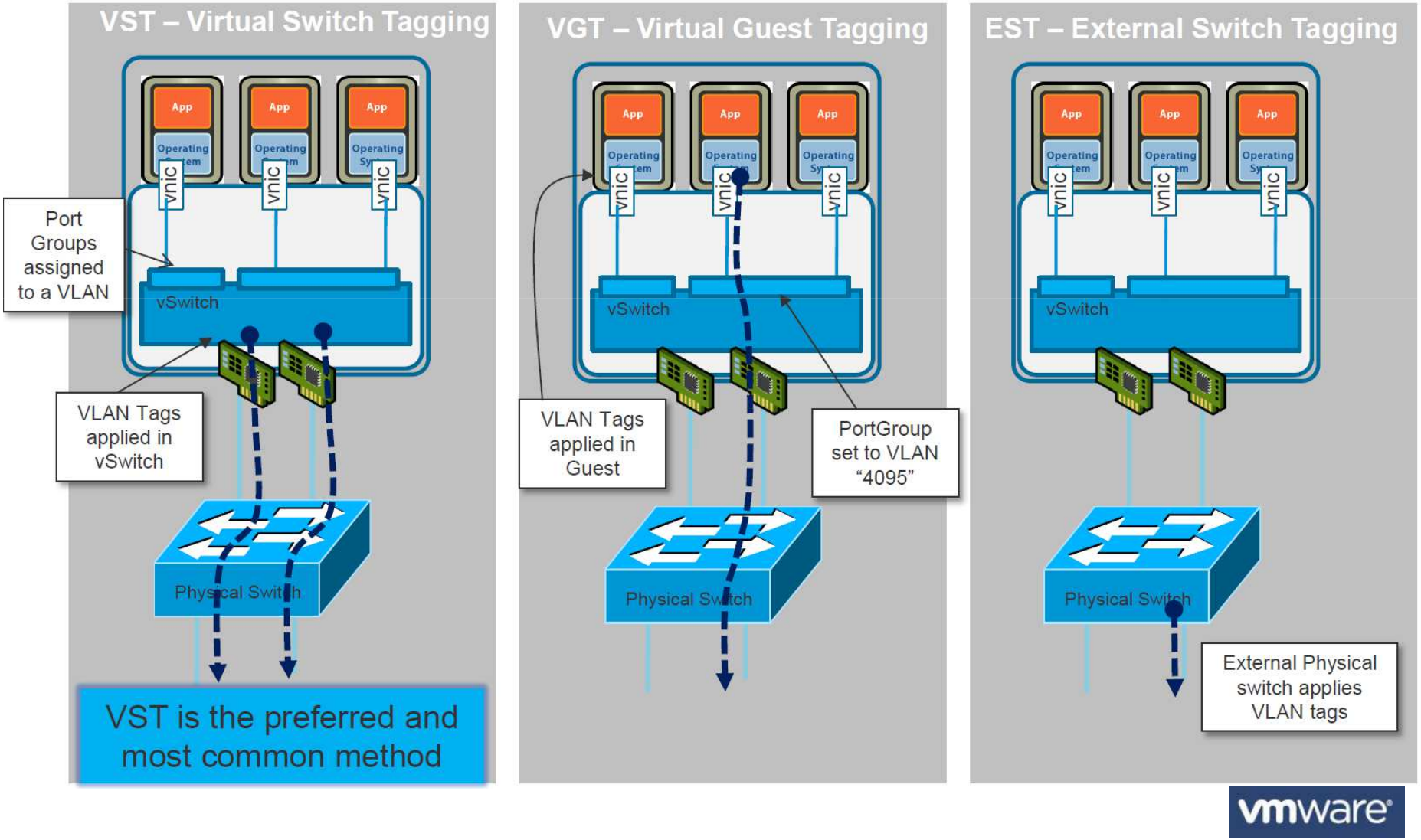
Toshiaki Makita. *Virtual switching technologies and Linux bridge*. NTT Open Source Software Center.

Open vSwitch vs Linux Bridge

Feature	Open vSwitch	Linux Bridge
MAC Learning Bridge	X	X
VLAN support (802.1Q)	X	Using ‘vlan’
Static Link Aggregation (LAG)	X	Using ‘ifenslave’
Dynamic Link Aggregation (LACP)	X	Using ‘ifenslave’
Support for MAC-in-IP encapsulation (GRE, VXLAN, ...)	X	VXLAN support in 3.7 kernel + Iproute2
Traffic capturing / SPAN (RSPAN with encap. Into GRE)	X	Using advanced Traffic Control
Flow monitoring (NetFlow, sFlow, IPFIX, ...)	X	Using ipt_netflow
External management interfaces (OpenFlow & OVSDB)	X	
Multiple-Table forwarding pipeline with flow-caching engine	X	

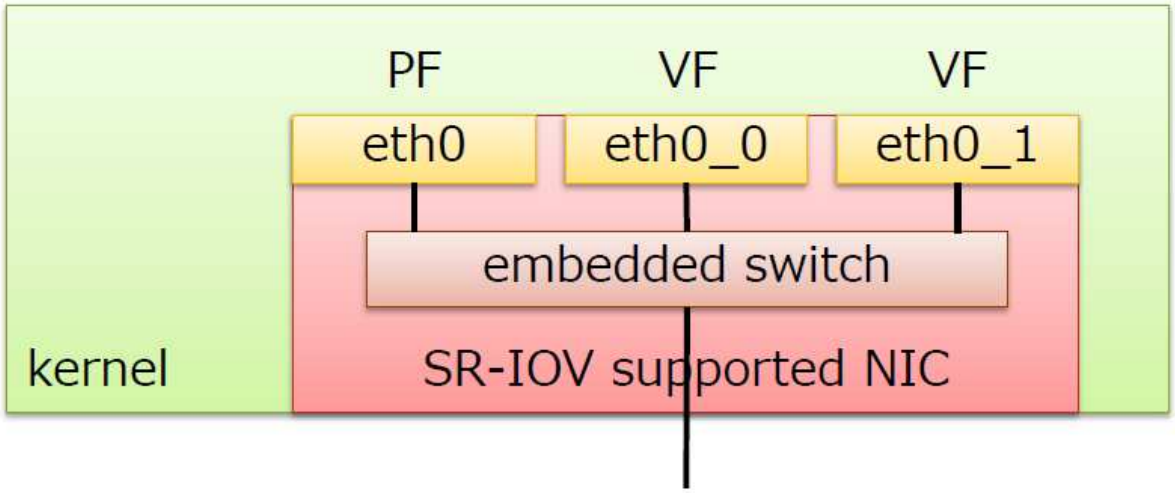
VLAN Tagging Options

- When a VM needs to participate in a VLAN spanning across several physical servers and switches, 802.1q VLAN tagging is needed
- Who tags packets ?



SR-IOV

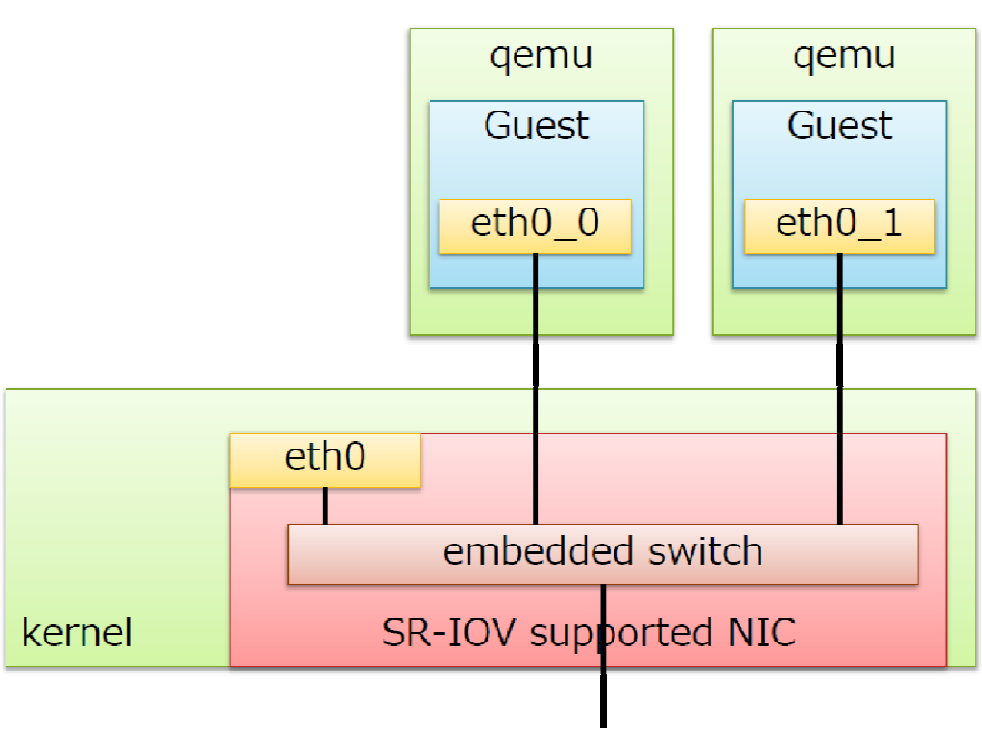
- Addition to PCI normal *physical function* (PF), allows to add lightweight *virtual functions* (VF)
- VF appears as a network interface
  - eth0_0, eth0_1, ...
- SR-IOV devices have switches in them that allow PF-VF/VF-VF communication
- DMA is used to copy packets directly into VM's memory space without CPU load
- In terms of performance, SR-IOV produces higher throughput than software switches with much less CPU load

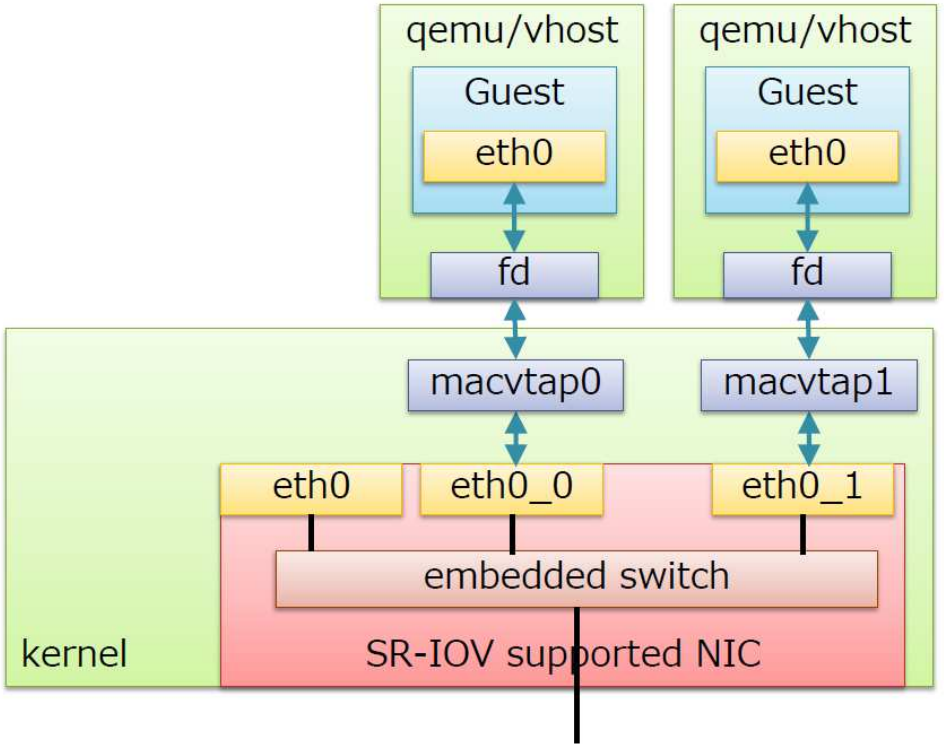


Toshiaki Makita. *Virtual switching technologies and Linux bridge*. NTT Open Source Software Center.

KVM with SR-IOV

- Two modes of operation:
  1. Use PCI-passthrough to attach VF to guest
  2. Use macvtap device (passthru)





Toshiaki Makita. ***Virtual switching technologies and Linux bridge.***
NTT Open Source Software Center.