

Mainframe Topics

Kapitel 4

Virtualisierung auf dem Mainframe

Introduction of Virtualization

Frank Heimes

Senior IT Architect & Technical Lead for Ubuntu on z
Canonical Ltd.

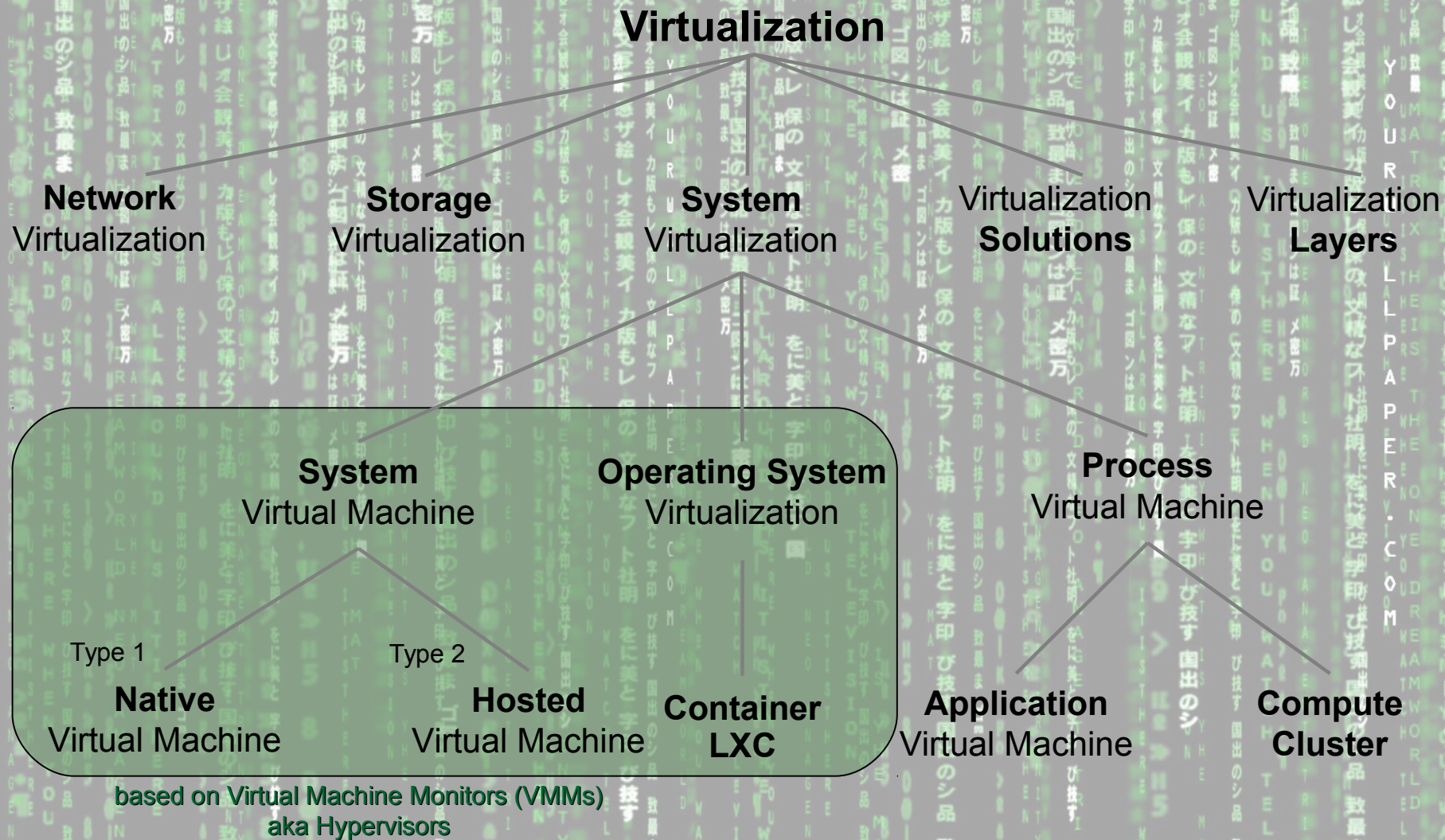
Agenda

- Virtualization – Say What?
- What is a Virtual Machine Image?
- Server Virtualization Approaches
- Open Virtual Machine Format - OVF
- VMware Virtual Infrastructure
- Hyper-V - Architecture
- KVM-Architecture
- Xen
- Current Dynamics of the open source virtualization
- The business value of virtualization

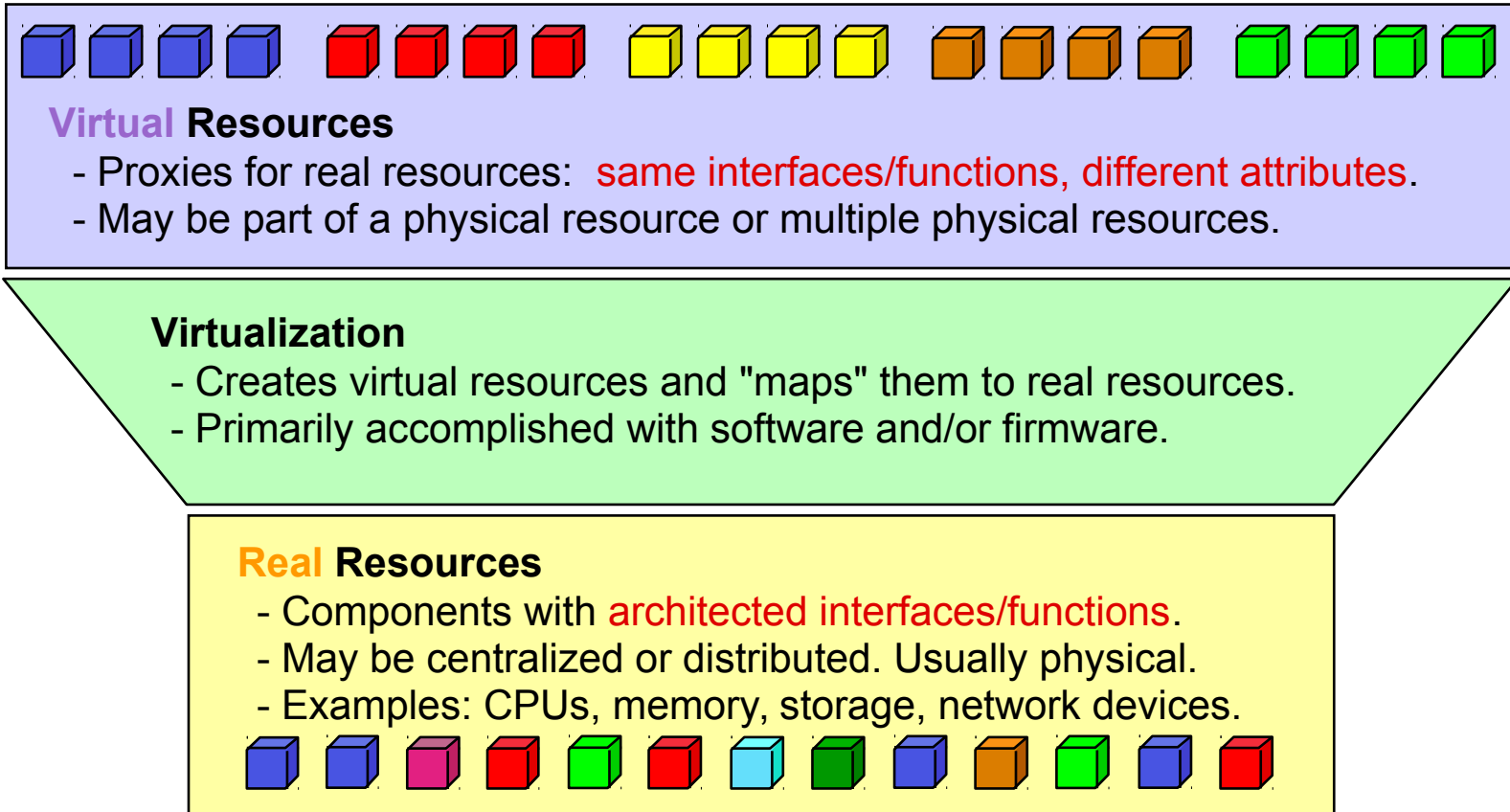
Learning targets

- Describe virtualization technologies
- Know important terms regarding virtualization
- Compare virtualization technologies and know advantages and disadvantages of them
- Describe the fundamental server virtualization approaches and assign appropriate examples
- Know the current open source virtualization solutions
- Know the business value of virtualization

Virtualization Taxonomy



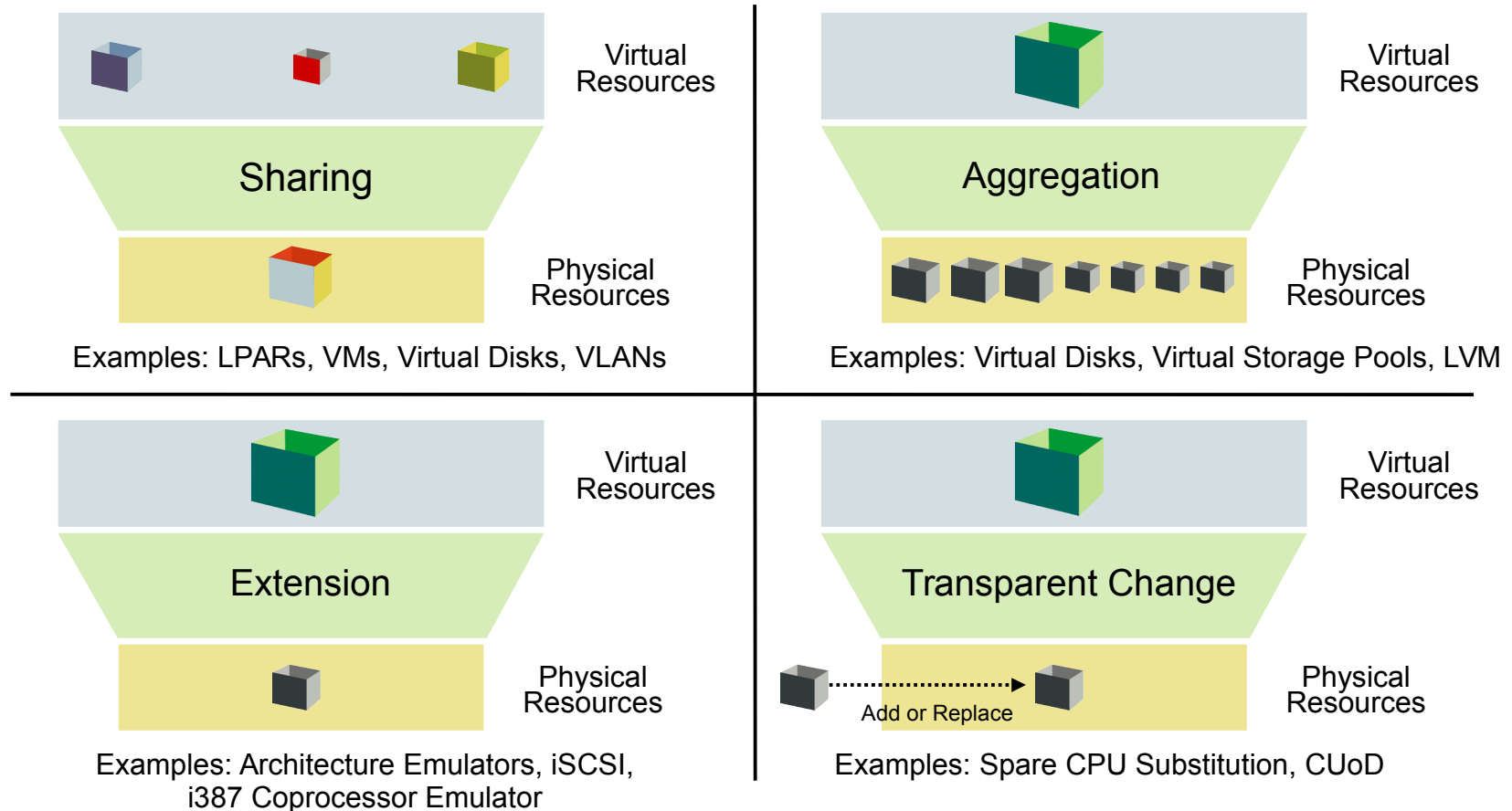
Virtualization – Say What?



- Separates presentation of resources to 'users' from actual resources
- Aggregates pools of resources for allocation to users as virtual resources

Virtualization Use Cases

Categorization of virtualization concepts

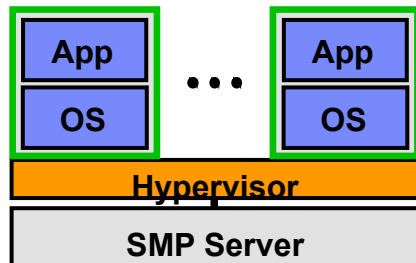


Type 1 and Type 2 Classification of Virtualization

Common distinction between the following two virtualization concepts:

- Type 1

- Bare-Metal / Native Hypervisor

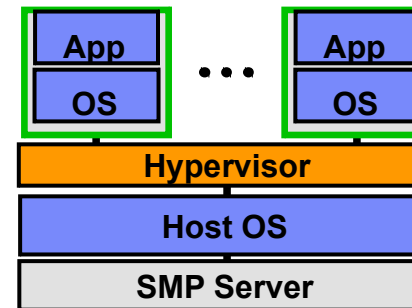


Hypervisor runs directly on the physical hardware (inside HW/FW or as OS)
e.g. IBM z/VM, VMware ESXi,
Linux KVM - if acting as HV only provides fine-grained timesharing of all resources

Type 1 hypervisor have their own hardware compatibility list (hcl)
(easy on z Systems, but hard on x86 or arm)

- Type 2

- Hosted Hypervisor



Hypervisor is build as application that runs on top of, and uses an Host OS
e.g. VMware Workstation, VirtualBox,
Linux KVM – if not exclusively used as HV, but also JVM, .Net CLR

Type 2 hypervisor are quite operating system dependent

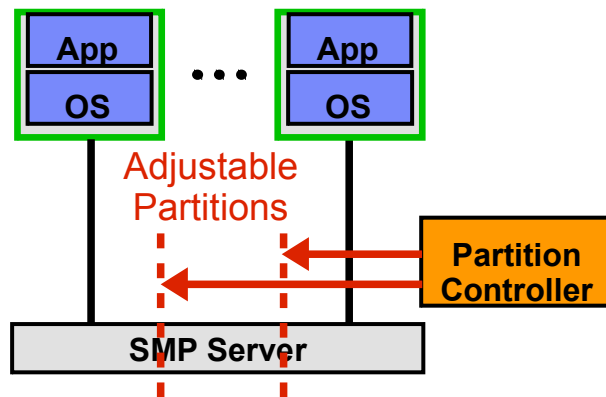
Hardware compatibility provided by the underlying OS (reuse of drivers)

Type '0' and Type '3' Classification

More virtualization concepts – NOT officially known as Type 0 and Type 3!

- **Type “0”**

- Hardware Partitioning



Server is subdivided into fractions, each of which can run an OS

Physical Partitioning:

(partly) hosted on the physical HW,
S/370, Sun Domains, HP nPar

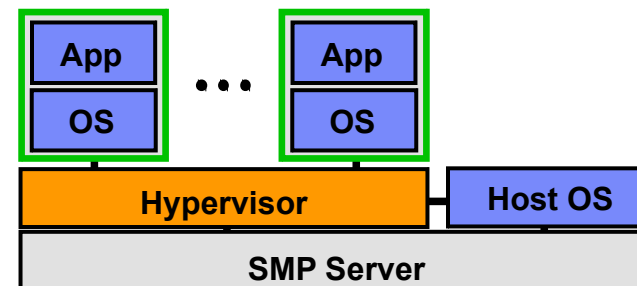
Logical Partitioning:

PR/SM \equiv LPAR, POWER LPAR, HP vPar

(SW Partitioning: AIX WPARS, BSD Jails,
SWsoft/Parallels Virtuozzo, OpenVZ,
Sun Containers, Linux Containers)

- **Type “3”**

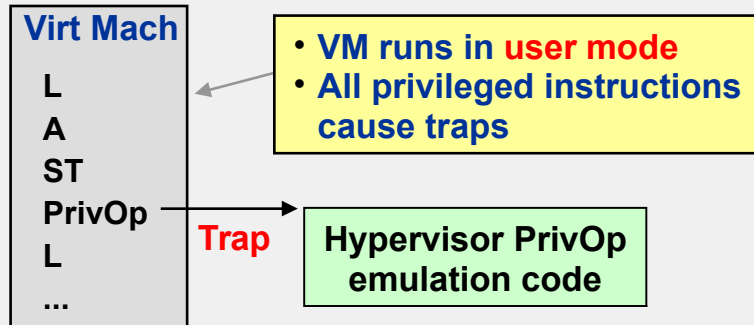
- “Hybrid” VMM



Hybrid VMMs use a bare-metal hypervisor for main tasks, but also a host operating system for special tasks like I/O virtualization
e.g. Microsoft Hyper-V (Viridian), Xen, VMware ESX Server, PowerVM + VIOS

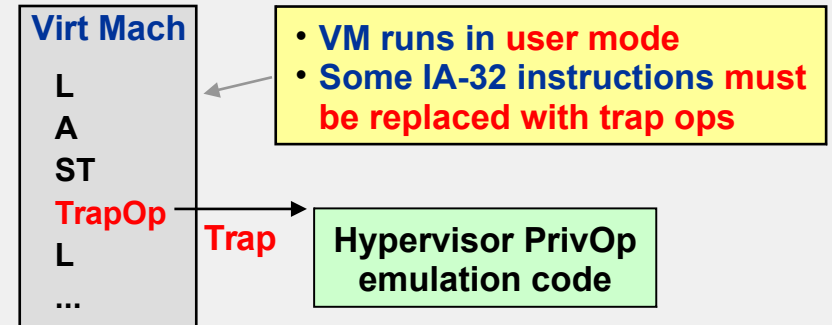
Hypervisor Implementation Methods

Trap and Emulate



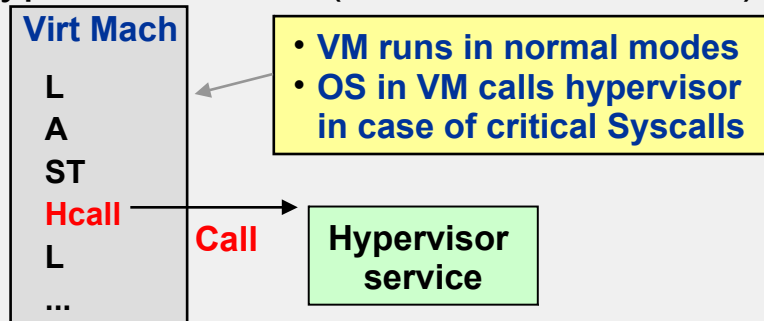
Examples CP-67, VM/370
 Benefits Runs unmodified OS
 Issues Substantial overhead

Translate, Trap, and Emulate



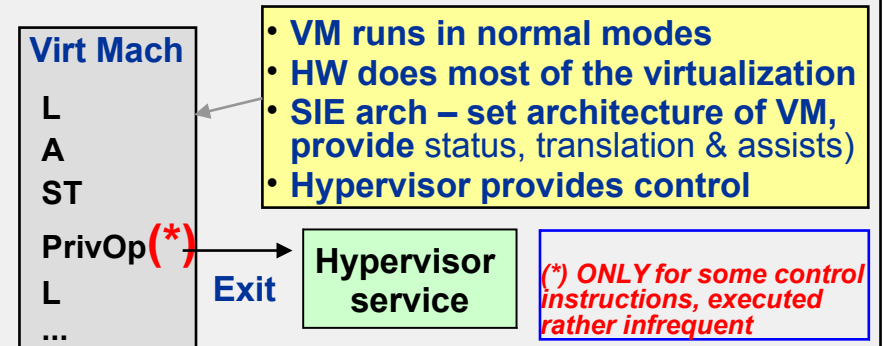
Examples VMware (today), Microsoft VS
 Benefits Runs unmodified, translated OS
 Issues May have some substantial overhead

Hypervisor Calls (“Paravirtualization”)



Examples POWER Hypervisor, Xen (today), HP Integrity VM
 Benefits High efficiency depending of Hypervisor code + eventual HW support
 Issues OS-Kernel must be modified to issue Hcalls. OS & Hypervisor levels must be in sync

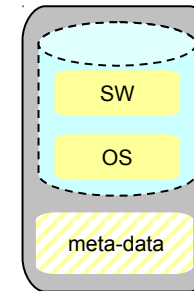
Direct Hardware Virtualization



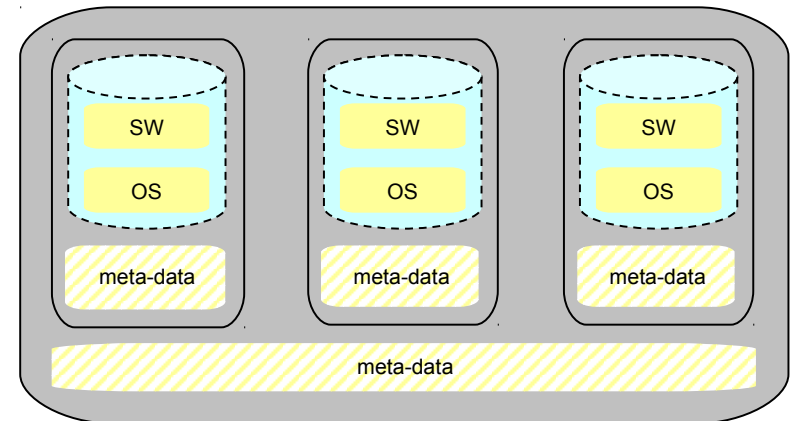
Examples PR/SM, z/VM (also use hypervisor calls for a some functional enhancements)
 Benefits Highest efficiency depending on HW/ucode support. Runs unmodified OS
 Issues Requires HW & ucode support

What is a Virtual Machine Image?

- Meta-data describing the required server resources
 - Number of CPUs (dedicated vs. shared)
 - Memory requirements
 - IO and network requirements
- Meta-data describing goals and constraints
 - Availability goals
 - Placement constraints
- Meta-data describing configuration variables
 - OS Configuration parameter – IP Address, etc.
 - Application Configuration parameters
- One or more disk images containing OS, middleware and other application software
- May be composition of virtual machine images
 - Virtual machine images making up a distributed application workload.
 - Includes additional meta-data scoped to the entire composition.



Virtual Machine Image



Virtual Machine Image Composition

KVM Overview

- 🔗 KVM (Kernel Virtual Machine) is a Linux kernel-based hypervisor
- 🔗 Developed and maintained by Avi Kivity / Qumranet, recently acquired by Red Hat
- 🔗 KVM turns the Kernel into a hypervisor by loading a kernel module and opening a device node. The main parts of KVM are:
 - Kernel module `kvm.ko`
 - Hardware specific modules
 - Device node `/dev/kvm` (to create/run VMs from userspace with a set of `ioctl()`s)
- 🔗 Virtual machines (or guests or domains) appear as normal Linux processes and integrate seamlessly into the rest of Linux
- 🔗 A VM has its own memory, that is separated from the user space process
- 🔗 Virtual CPUs are not scheduled on it's own (vCPUs are realized as Linux threads, and are still scheduled by the Linux Kernel process scheduler)
- 🔗 In full virtualization mode it's possible to run multiple unmodified guest OSes in parallel, with each having private virtual hardware (network, disk, graphics etc.)
- 🔗 Exploits 'SIE' hardware instruction on z Systems

KVM Design Considerations

🕒 Main hypervisor requirements are:

- Architecture support
- Memory management
- CPU scheduler
- I/O stack and scheduler (incl. TCP/IP)
- Device drivers (file-systems, disks (ide, scsi, fc, ...), nics, etc.)
- Power Management
- Security features
- ...

🕒 But the Linux operating system has already world-class support for all this ?!
(99%)

🕒 So do not reinvent the wheel !

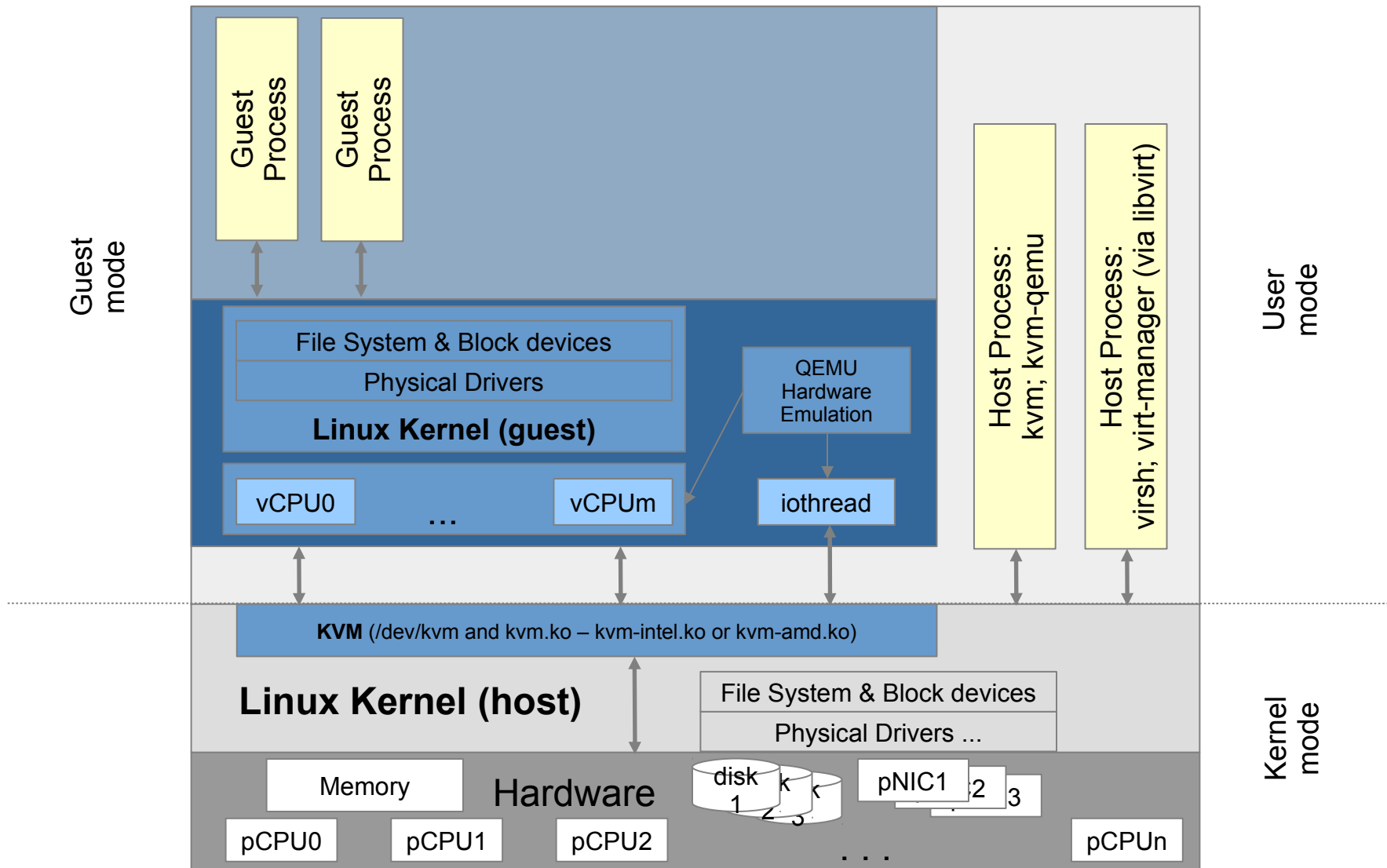
🕒 Reuse existing Linux code (Kernel and OS) as much as possible.

🕒 Focus on virtualization only – integrate and implement the relevant topics like the (hardware) virtualization support, traps, translation or emulation if required.

🕒 Leave other things to their respective developers.

🕒 Benefit from all semi-related advances of the Linux OS.

QEMU/KVM Component Diagram



Qemu/KVM Host Processes

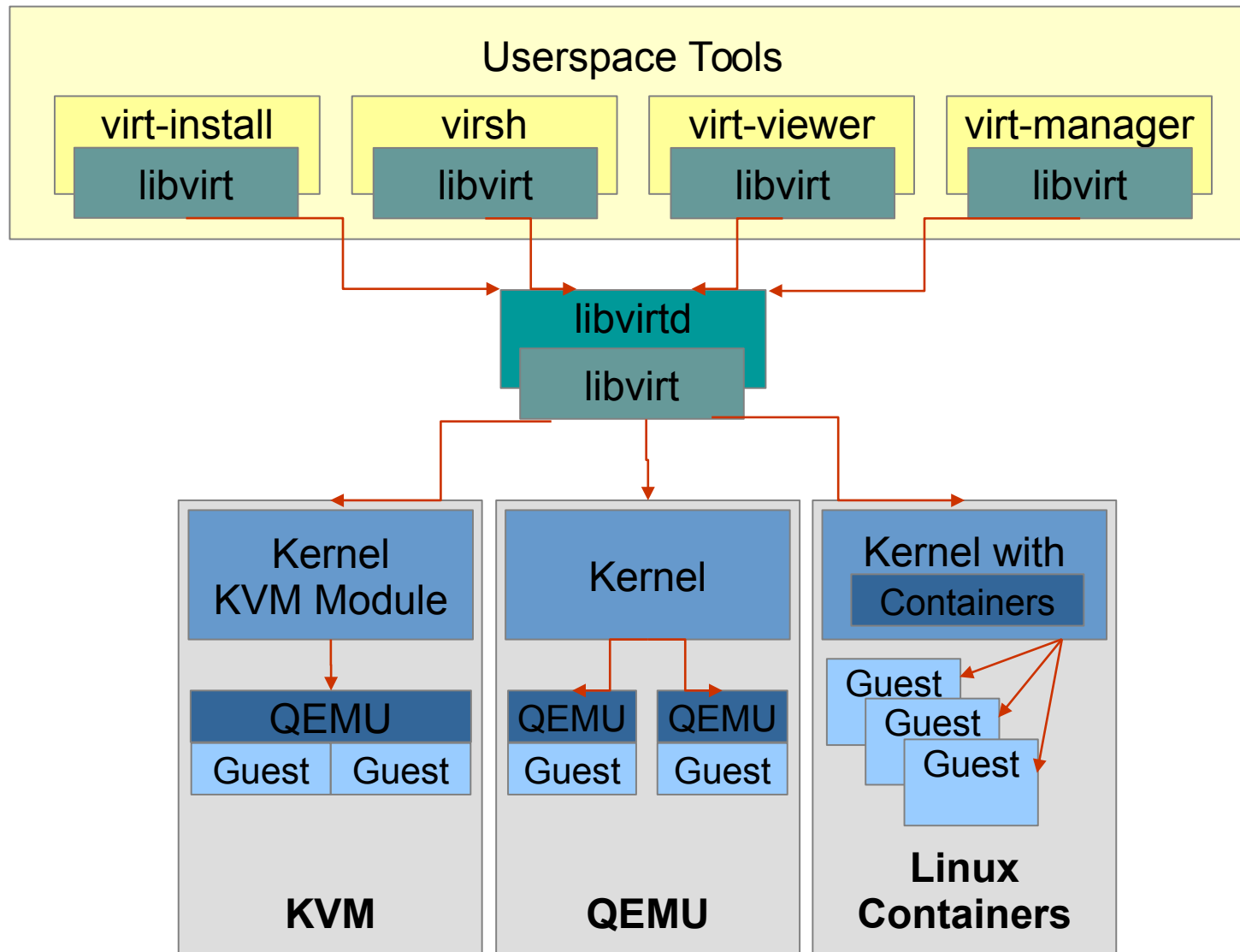
```
top - 22:47:19 up 1 min,  1 user,  load average: 1.67, 1.15, 0.46
Mem:   10064M total,    839M used,    9224M free,     9M buffers
Swap:   2039M total,     0M used,    2039M free,    615M cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6686	root	20	0	662m	134m	130m	S	0.0	1.3	0:03.59	/usr/bin/qemu-kvm -name zlin175 -S -machine s390-ccw-virtio,
6769	root	20	0	650m	130m	125m	S	0.0	1.3	0:03.43	/usr/bin/qemu-kvm -name zlin176 -S -machine s390-ccw-virtio,
4159	haldaemo	20	0	12960	8084	2284	S	0.0	0.1	0:00.77	/usr/sbin/hald --daemon=yes
5628	root	20	0	110m	7216	4944	S	0.0	0.1	0:00.30	/usr/sbin/libvirtd --daemon --config /etc/libvirt/libvirtd.c
5578	root	20	0	6464	4508	272	S	0.0	0.0	0:00.00	/sbin/haveged -w 1024 -v 1
6585	root	20	0	11444	4028	3156	S	0.0	0.0	0:00.01	sshd: root@pts/0
4162	root	20	0	9032	3232	1920	S	0.0	0.0	0:00.00	/usr/sbin/console-kit-daemon
6588	root	20	0	6100	3096	1736	S	0.0	0.0	0:00.01	-bash

```
,accel=kvm,usb=off -m 512 -smp 2,sockets=2,cores=1,threads=1 -uuid abd379e5-50fe-03e4-af46-7dbf7ed7ce7f -nographic -n
,accel=kvm,usb=off -m 500 -smp 1,sockets=1,cores=1,threads=1 -uuid 30b57065-9eaa-682f-5b51-27dc64184973 -nographic -n

conf --listen
```

libvirt Component Diagram



Let's talk about...

Business Value!

Virtualization Advantages

- Increased efficiency and consolidation
 - Better hardware utilization
 - Less energy consumption / Less heat generation
 - Less maintenance costs / contracts for hardware
 - Less space allocation
 - Rapid deployment of new systems / Dynamic Provisioning / System Transition
 - Clear system landscape / Simplified systems management
 - Workload Management
- Mixed-OS environments
- Legacy applications
- Virtual Hosting
- Business Continuity
- High Availability
- Disaster Recovery
- Security / Isolation / Encapsulation
- RAS

Virtualization Disadvantages

Effort (overhead) for guest *and* host operating system or hypervisor required

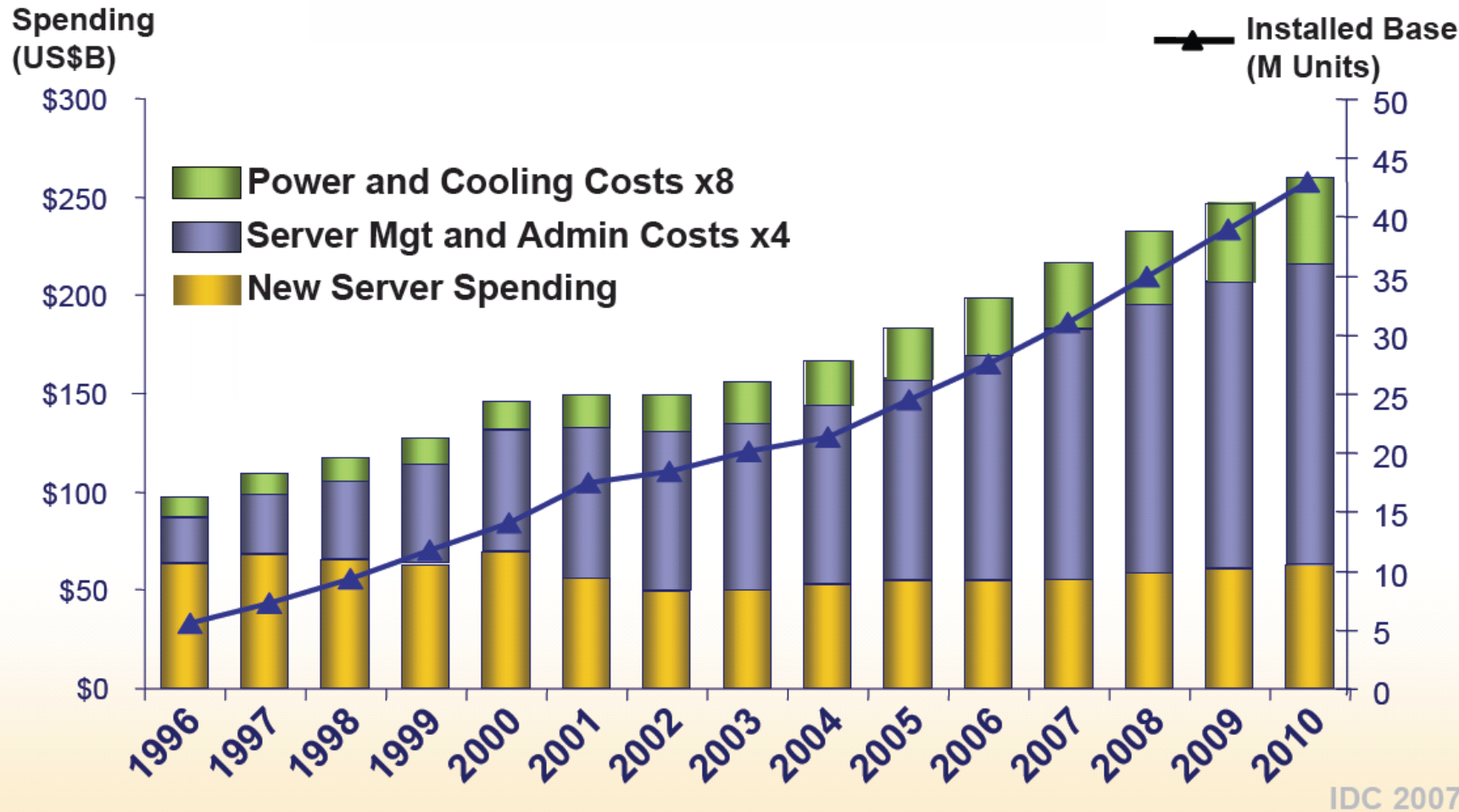
Hosted hypervisor may require additional (host) operating system (license)

Virtual Root Kits

- VMBRK: Virtual Machine-based Root Kits
 - needs to be “installed”
 - needs a system restart
 - remains persistent
- HVBRK: Hardware Virtualization-based Root Kits
 - runs in memory
 - difficult to detect
 - not persistent, yet
- Rootkits in System Management Mode (SMM) memory via CPU Cache Poisoning (Intel)

What are data centres facing today?

Explooding numbers of systems!



Thank You

धन्यवाद

Hindi

Hindi

English

ขอบพระคุณ

Thai

Спасибо

Russian

Gracias

Spanish

多謝

Traditional Chinese

شكراً

Arabic

Obrigado

Brazilian Portuguese

Danke

German

Grazie

Italian

多谢

Simplified Chinese

Merci

French

நன்றி

Tamil

Tamil

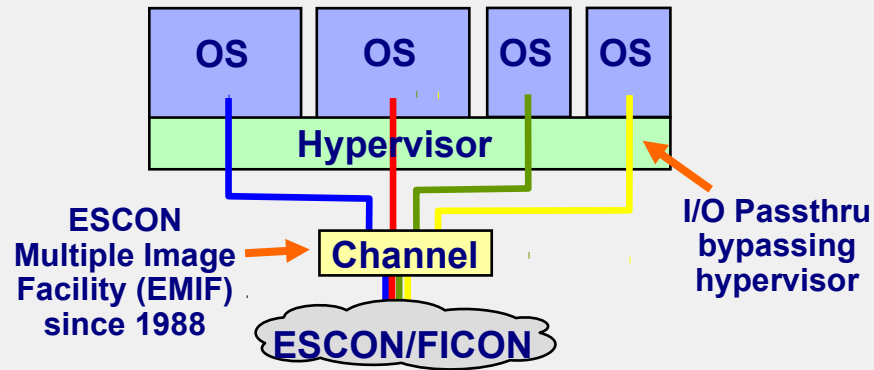
ありがとうございました

Japanese

감사합니다

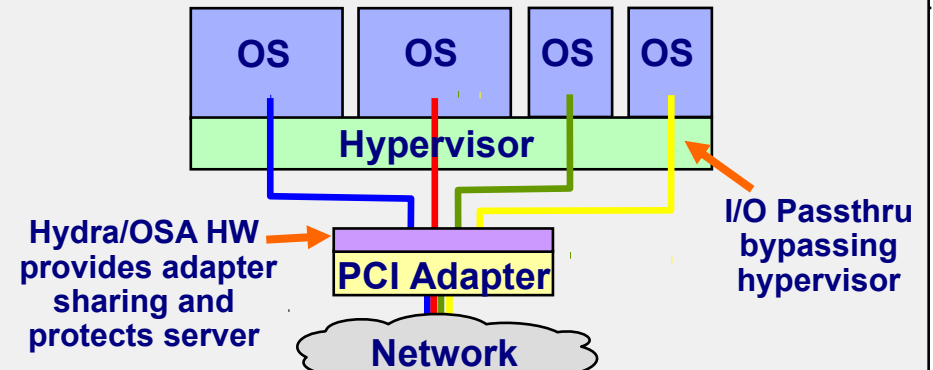
Korean

System z Native Adapter & Network Virtualization



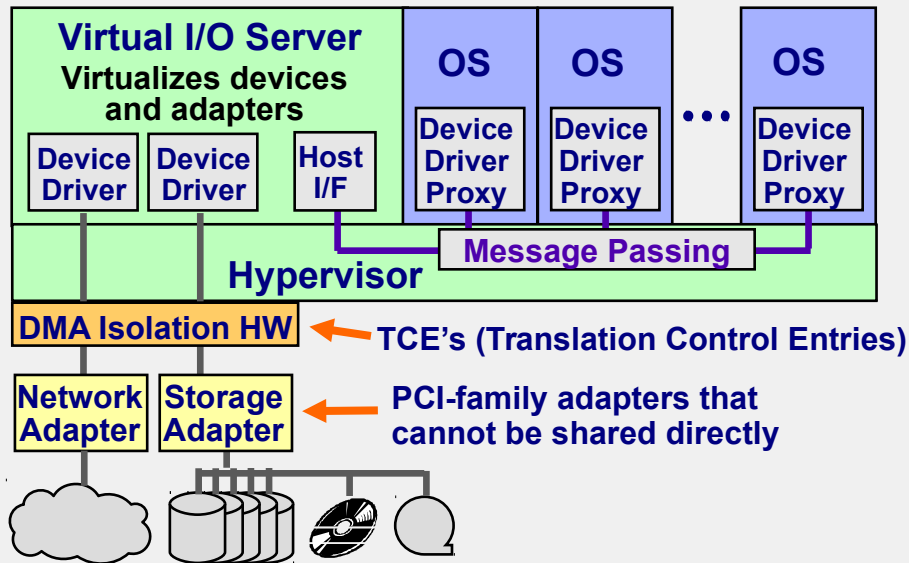
- ESCON channels and network support efficient sharing

System z PCI Adapter Virtualization



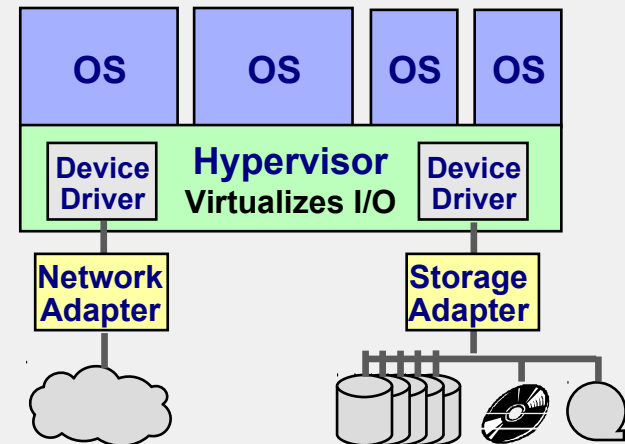
- PCI-family I/O adapters cannot be shared directly

System p Device and Adapter Virtualization



- Firmware (VIOS) provides I/O sharing
- Hardware (TCE's) provide I/O isolation

VMware PCI Adapter Virtualization



- I/O virtualization and device drivers are part of hypervisor reducing overall system availability
- Failure of I/O adapter or device driver can cause system outage or data corruption