9th Slide Set Cloud Computing

Prof. Dr. Christian Baun

Frankfurt University of Applied Sciences
(1971–2014: Fachhochschule Frankfurt am Main)
Faculty of Computer Science and Engineering
christianbaun@fb2.fra-uas.de

Agenda for Today

- Virtualization
  - Benefits of virtualization
  - Drawbacks and limitations of virtualization
- Concepts of virtualization
  - Partitioning
  - Hardware emulation
  - Application virtualization
  - Full virtualization (Virtual Machine Monitor)
  - Paravirtualization (Hypervisor)
  - Hardware virtualization
  - Operating system-level virtualization / Container / Jails
  - Storage virtualization (SAN)
  - Network virtualization (VLAN)

Virtualization – Fundamentals

- By using **virtualization**, the resources of a computer system can be split and used by multiple independent operating system instances
- Several fundamentally different approaches and technologies exist to implement virtualization
- Each **virtual machine** (VM)...
  - behaves like any other computer, with own components
  - runs inside an isolated environment on a physical machine
- Inside a VM, an operating system with applications can be installed, exactly like on a physical computer
  - The applications do not notice that they are located inside a VM
- Requests of the operating system instances are captured by the virtualization software and converted for the existing physical or emulated hardware
  - The VM itself does not become aware of the virtualization layer between itself and the physical hardware

History of Virtualization

- Virtualization is not a new concept
  - Introduced in the 1960s by IBM for mainframes
- 1970/71: IBM introduced the Virtual Machine Facility/370 (VM/370)
  - On this platform, multi-user operation is implemented by using multiple single-user mode instances, which are executed in virtual machines
  - Each VM is a complete duplicate of the underlying physical hardware

Sources

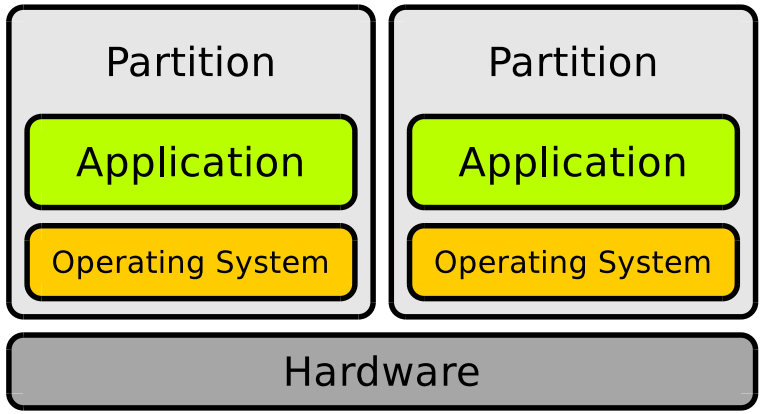
- Creasy RJ. **The origin of the VM/370 time-sharing system**. IBM Journal of Research and Development 25 (1981), No. 5, 483–490
- Amit Singh. **An Introduction to Virtualization**. 2004 `http://www.kernelthread.com/publications/virtualization/`

Virtualization Concepts

- Different virtualization concepts exist:
  - Partitioning
  - Hardware emulation
  - Application virtualization
  - Full virtualization (Virtual Machine Monitor)
  - Paravirtualization (Hypervisor)
  - Hardware virtualization
  - Operating system-level virtualization / Container / Jails
  - Storage virtualization (SAN)
  - Network virtualization (VLAN)
  - ...

Partitioning

- If partitioning is used, the total amount of resources can be split to create subsystems of a computer system
  - Each subsystem may contain an executable operating system instance
  - Each subsystem can be used like an independent computer system



- The resources (CPU, main memory, storage. . .) are managed by the **firmware** of the computer and assigned to the VMs
- Partitioning is used, e.g. in IBM mainframes (zSeries) and midrange systems (pSeries) with Power5/6/7 CPUs
  - Resource allocation is possible during operation without having to restart
  - On a modern mainframe computer several hundred to thousands of Linux instances to operate simultaneously
- Modern CPUs only support the partitioning of the CPU itself and not of the entire system (Intel Vanderpool, AMD Pacifica)
  - Partitioning is not used for desktop environments

Partitioning Example – Watson (1/2)

- February 2011: *Watson* wins the Quiz *Jeopardy Challenge* in the U.S.
  - Watson is a cluster of 90 IBM Power 750 servers with 2,880 Power7 CPU cores (each with 8 cores per CPU) and 16 TB RAM

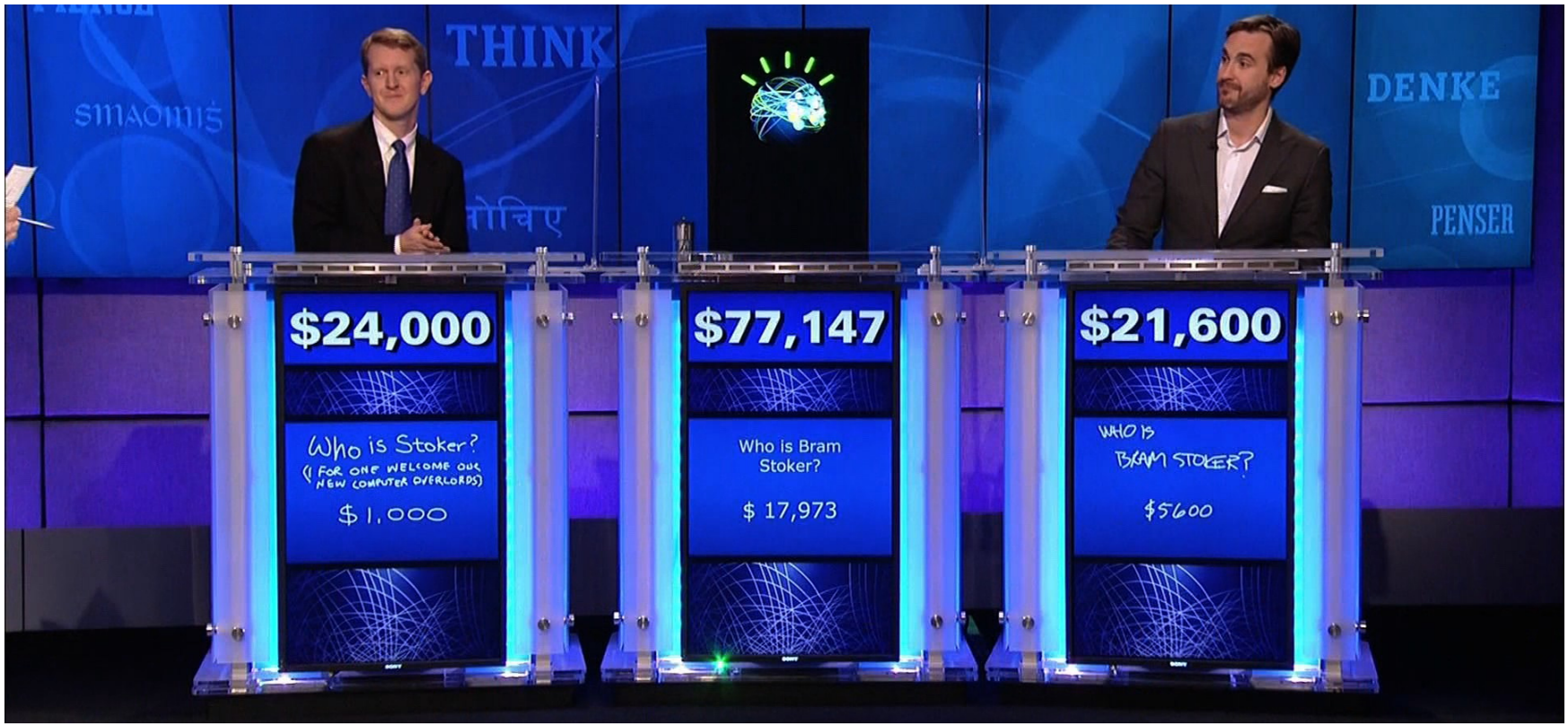


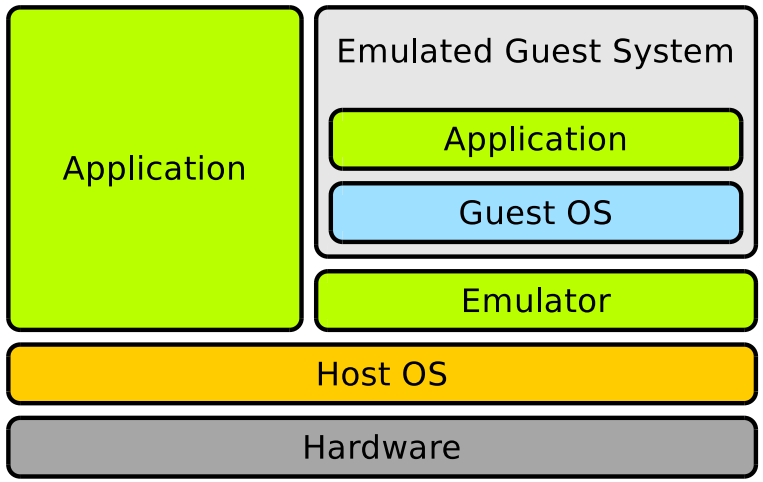
Image source: `http://www.cnmeonline.com/wp-content/uploads/2013/03/IBM-Watson.jpg`

Partitioning Example – Watson (2/2)

- Partitions can be created at each one of the 90 nodes
  - Each partition may contain an AIX, Linux or IBM i (formerly OS/400)
  - The partitions are independent installations
    - Each partition can contain a different operating system
- On each node runs a *POWER Hypervisor*
  - It controls the hardware access
- Since Power6, running partitions can be relocated without interruption from one physical server to another one ($\Longrightarrow$ Live Partition Mobility)
- Partitions can share main memory ($\Longrightarrow$ Active Memory Sharing)
  - Active Memory Expansion is able to compress storage pages
    - Depending on the application, compression is faster compared with relocating or swapping

Hardware Emulation

- Emulation duplicates the **complete hardware** of a computer system, to allow the execution of an **unmodified operating system**, which is designed for a **different hardware architecture** (CPU)
  - Exception: Wine
    - Wine does not emulate hardware, but only the interfaces of the Windows operating system



- Drawbacks of emulation:
  - Development is very expensive
  - Performance is low compared with virtualization
- Important distinction: **emulation** $\neq$ **virtualization**
- Some emulators: Bochs, QEMU, PearPC, Wabi, DOSBox, Microsoft Virtual PC (the MacOS X/PowerPC version is a x86 emulator)

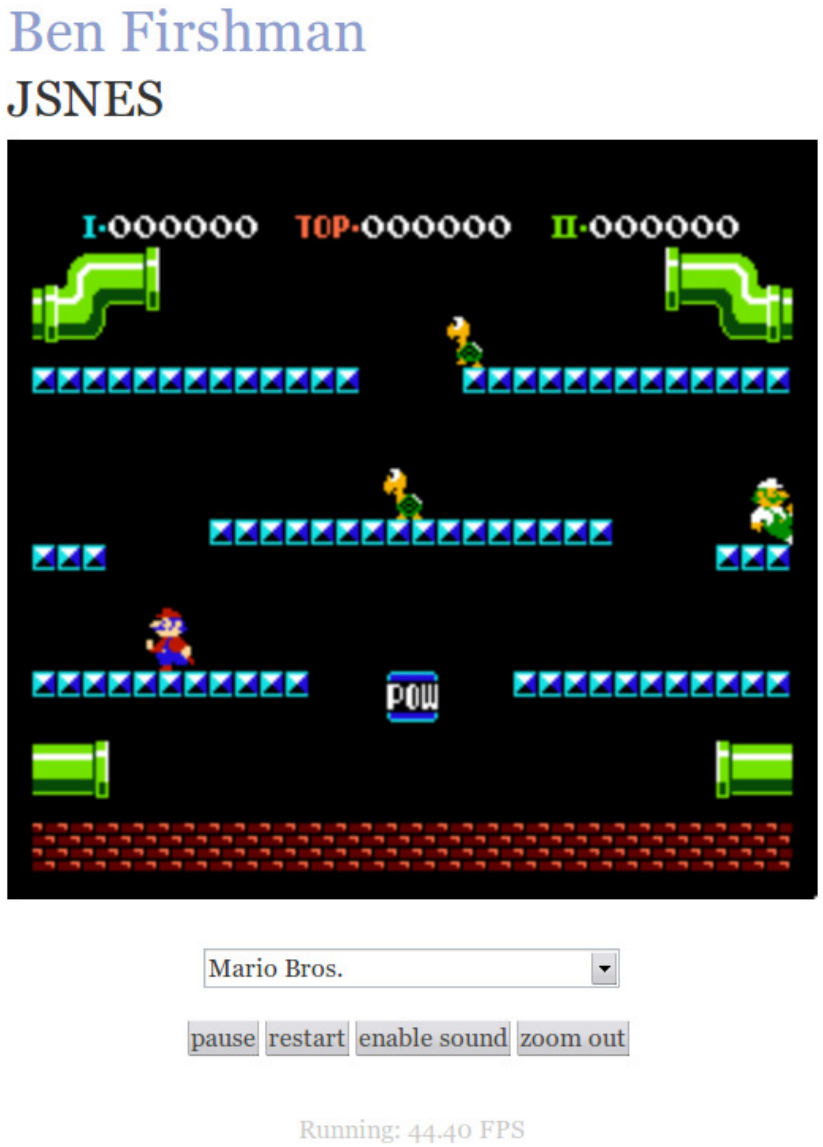
Selection of Emulators

Name	License	Host	Emulated architecture	Guest system
Bochs v2.3.6	LGPL	Linux, Solaris, MacOS, Windows, IRIX, BeOS	x86, AMD64	Linux, DOS, BSD, Windows, BeOS
QEMU v0.9.0	GPL	Linux, BSD, Solaris, BeOS, MacOS-X	x86, AMD64, PowerPC, ARM, MIPS, Sparc	Linux, MacOS-X, Windows, BSD
DOSBox v0.72	GPL	Linux, Windows, OS/2, BSD, BeOS, MacOS-X	x86	DOS
DOSEMU v1.4.0	GPL	Linux	x86	DOS, Windows bis 3.11
PearPC v0.4.0	GPL	Linux, MacOS-X Windows	PowerPC	Linux, MacOS-X, BSD
Baseilisk II v0.9-1	GPL	Linux, various UNIX, Windows NT4, BeOS, Mac OS, Amiga OS	680x0	MacOS $\leq$ 8.1
Wabi v2.2	proprietary	Linux, Solaris	x86	Windows 3.x
MS Virtual PC v7	proprietary	MacOS-X	x86	Windows, (Linux)
M.A.M.E. v0.137	MAME-Lizenz	Linux, Windows, DOS, BeOS, BSD, OS/2	various Arcade	various Arcade
SheepShaver	GPL	Linux, MacOS-X, BSD Windows, BeOS	PowerPC, 680x0	MacOS 7.5.2 bis MacOS 9.0.4
Hercules 3.07	QPL	Linux, MacOS-X, BSD Solaris, Windows	IBM mainframes	IBM System/360, 370, 390

- The table is not complete!
- Many more emulators exist

Example of a current Emulator - JSNES

- JSNES emulates the Nintendo Entertainment System (NES)
- The emulator is implemented in JavaScript and executes in the browser
- `http://fir.sh/projects/jsnes/`
- Free Software (GPLv3)



Latest Development: Browser emulates PC – jslinux

http://www.webmonkey.com/2011/05/yes-virginia-that-is-linux-running-on-javascript/

Date: May 18th 2011
Author: Scott Gilbertson

JavaScript never seems to get any respect. It's not a real programming language, detractors complain, it's just some script language that runs in the web browser. We're not sure what makes JavaScript less „real" to some, but thanks to today's web browsers, JavaScript has become a very powerful language. Powerful enough to run Linux in your web browser. French developer Fabrice Bellard has built a **JavaScript-based x86 PC emulator capable of running Linux inside a web browser**.

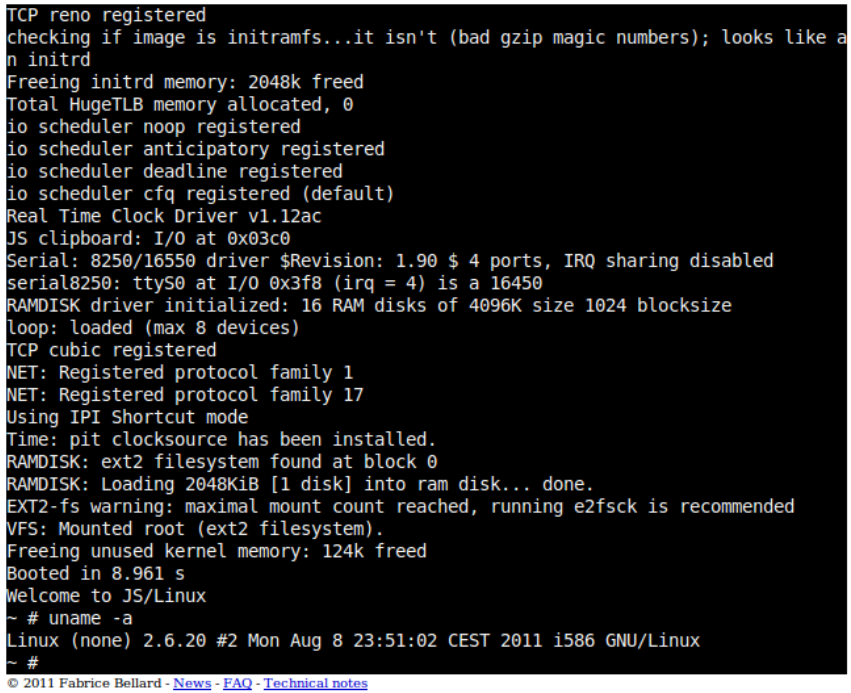


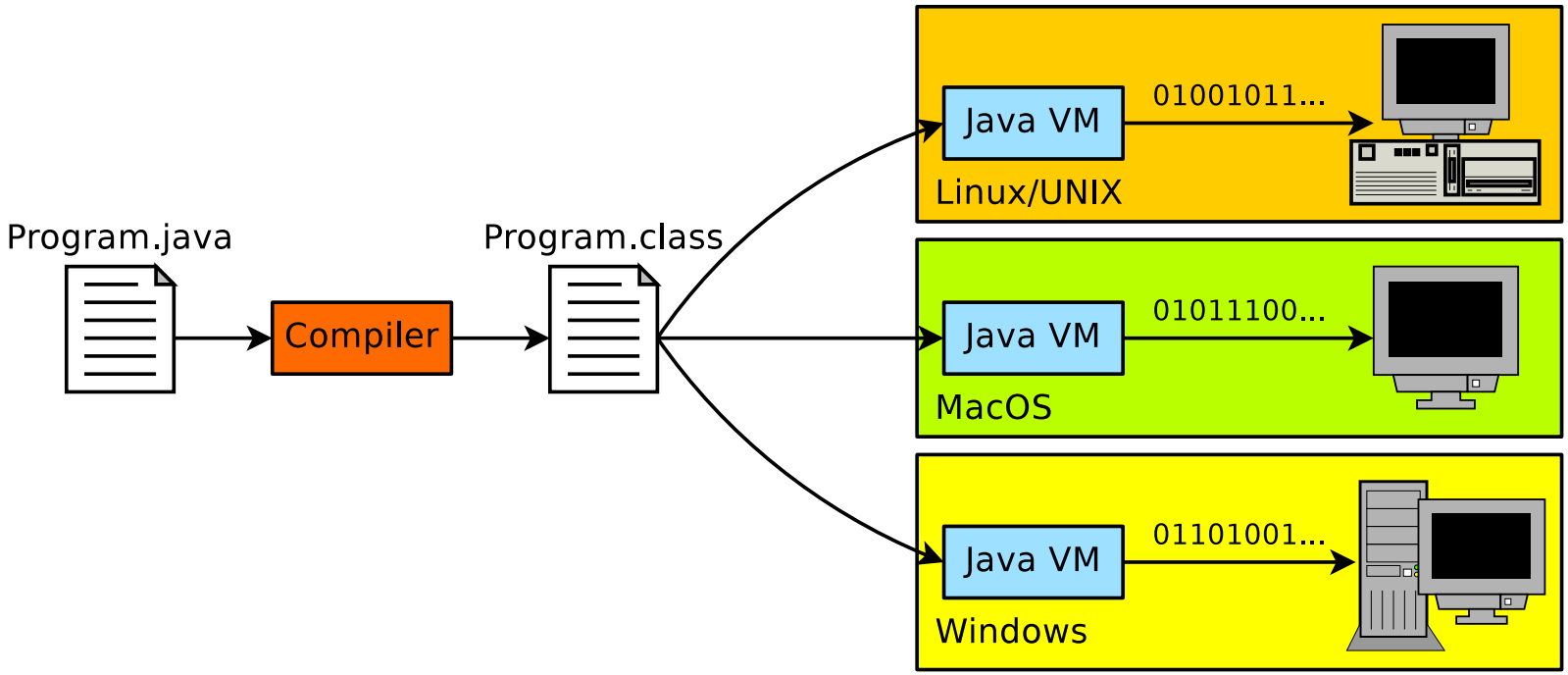
Image Source: http://bellard.org/jslinux/

If you'd like to try it out, point Firefox 4 or Chrome 11 to the demo page. Keep in mind that this is just Linux, no X Window or other graphical interface, just the command line, a small C compiler and QEmacs, Bellard's emacs clone. Still, it's really Linux, really running in your web browser, really using JavaScript to emulate hardware.

Application Virtualization

- Applications are executed inside a virtual environment, which uses local resources and provides all the components, which are required by the application
  - The VM is located between the executed application and the operating system
- Popular example: Java Virtual Machine (JVM)
  - The JVM is the part of the Java Runtime Environment (JRE), which executes the Java bytecode
  - The JVM is for Java programs the interface to the computer system and its operating system
- Advantage: Platform independence
- Drawback: Reduced performance, compared with native execution

Principle of the Java Virtual Machine (JVM)



- The compiler `javac` compiles source code into architecture-independent `.class` files, which contain bytecode, that can be executed in the Java VM
- The program `java` launches a Java application inside a Java VM

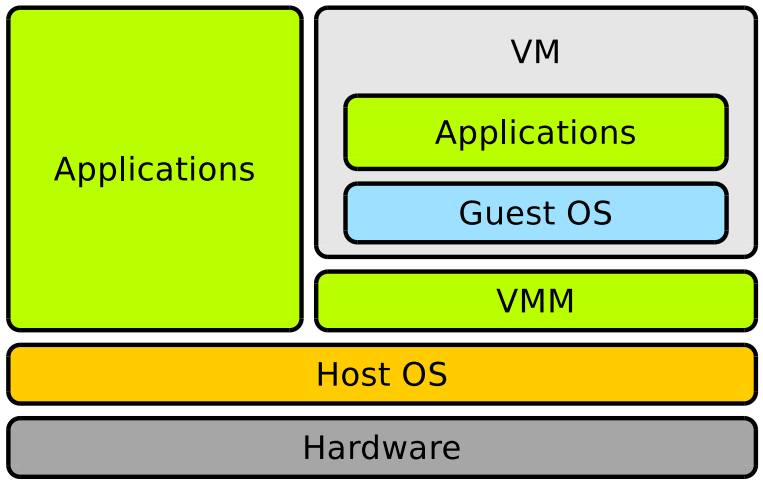
VMware ThinApp

http://www.vmware.com/products/thinapp/

- Further example of application virtualization: VMware ThinApp
  - Until 2008, the software was named Thinstall
- Packs Windows applications into single `.exe` files
- The application becomes portable and can be used without local installation
  - Applications can, e.g. be executed from an USB flash memory drive
- No entries are inserted into the Windows registry and no environment variables or DLL files are created on the system
- User preferences and created documents are stored inside a separate sandbox
- Drawback: The software only supports Microsoft Windows

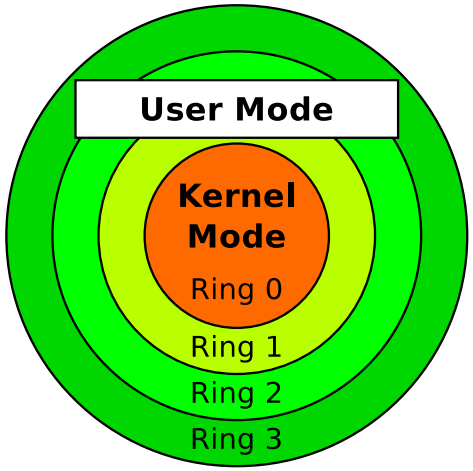
Full Virtualization (1/3)

- Full virtualization software solutions provide each VM a complete virtual PC environment, including an own BIOS
  - Each guest operating system gets its own VM with virtual resources (e.g. CPU, main memory, storage drives, network adapters)
- A **Virtual Machine Monitor** (VMM) is used
  - The VMM is also called **Type-2 hypervisor**
  - The VMM runs *hosted* as an application in the host operating system
  - The VMM distributes hardware resources to VMs
- Some hardware components are emulated, because they are not designed for the concurrent access from multiple operating systems
  - Example: Network adapters
  - The emulation of popular hardware avoids driver issues



Virtualization Basics of the x86 Architecture (1/2)

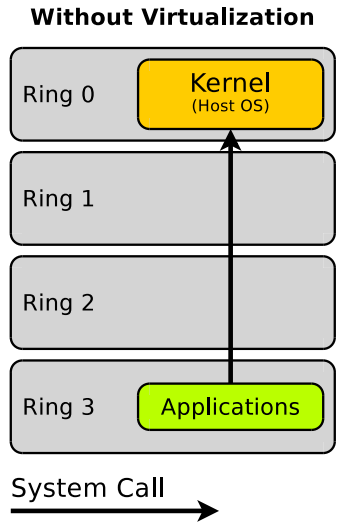
- x86-compatible CPUs contain 4 privilege levels
  - Objective: Improve stability and security
  - Each process is assigned to a ring permanently and can not free itself from this ring



Implementation of the privilege levels

- The register CPL (Current Privilege Level) stores the current privilege level
- Source: Intel 80386 Programmer's Reference Manual 1986
  `http://css.csail.mit.edu/6.858/2012/readings/i386.pdf`

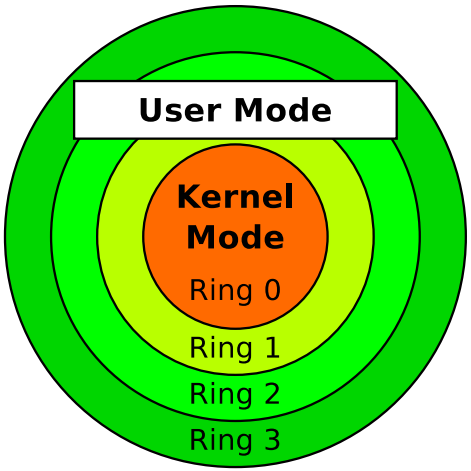
- In ring 0 (= **kernel mode**) runs the kernel
  - Processes in this ring have full access to the hardware
  - The kernel can address physical memory ($\Longrightarrow$ Real Mode)
- In ring 3 (= **user mode**) run the applications
  - Processes in this ring can only access virtual memory ($\Longrightarrow$ Protected Mode)



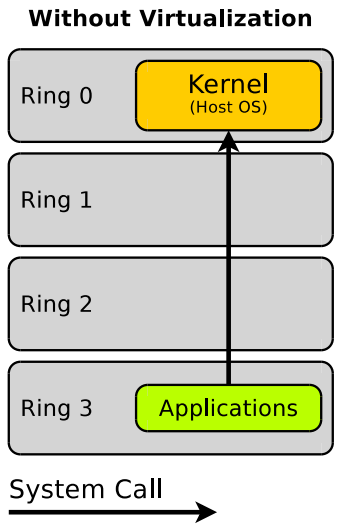
Virtualization Basics of the x86 Architecture (2/2)

Modern operating systems only use 2 privilege levels (rings)

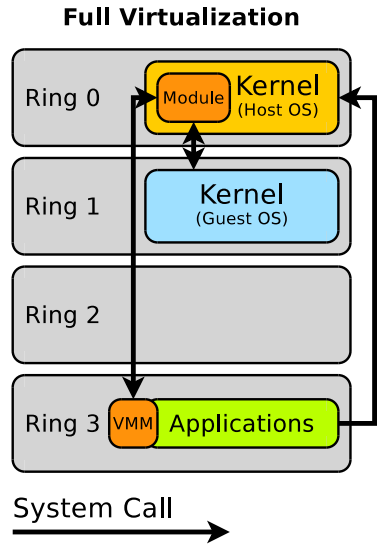
- Reason: Some hardware architectures (e.g. Alpha, PowerPC, MIPS) support only 2 privilege levels
- Exception: OS/2 uses ring 2 for applications, which are allowed to access hardware and input/output interfaces (e.g. graphics drivers)



- If a user-mode process must carry out a higher privileged task (e.g. access hardware), it can tell this the kernel via a **system call**
  - The user-mode process generates an exception, which is caught in ring 1 and handled there

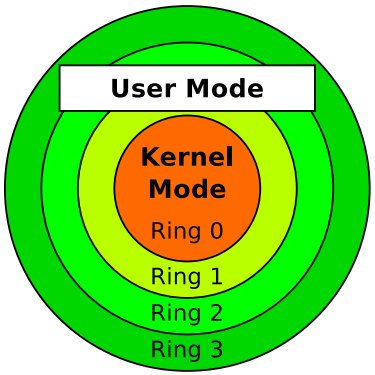


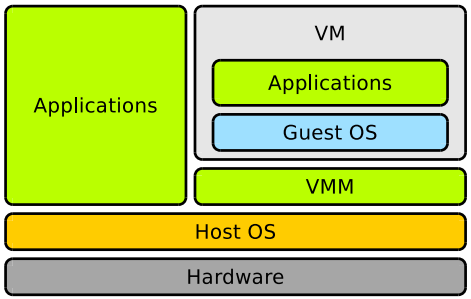
Full Virtualization (2/3)



- Full virtualization makes use of the fact, that x86 systems typically use only 2 privilege levels
  - The VMM runs together with the applications in ring 3
  - VMs are located in the less privileged ring 1

- The VMM contains for every exception a treatment, which catches, interprets and executes privileged operations of guest operating systems





- VMs can only access the hardware via the VMM
  - This ensures controlled access to shared system resources

Full Virtualization (3/3)

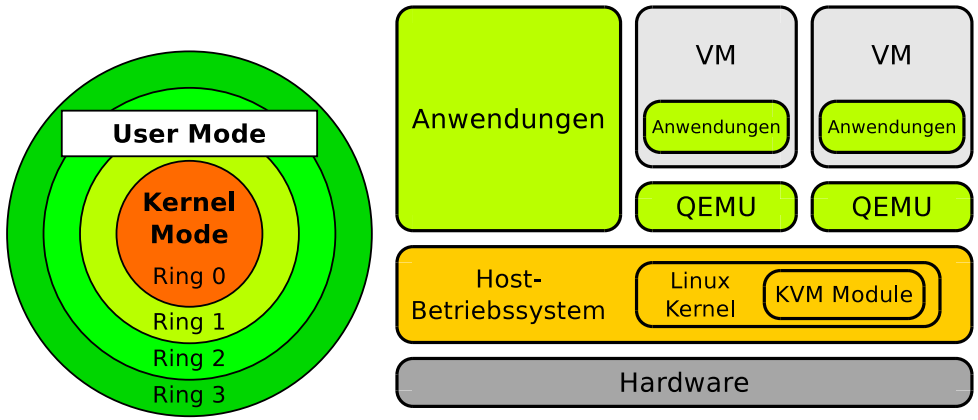
- Advantages:
  - Only few modifications in the host and guest operating systems are required
  - Access to the main resources is only passed through
    $\Longrightarrow$ guest operating systems operate with almost native processing speed
  - Each guest operating system has its own kernel
    $\Longrightarrow$ high degree of flexibility
- Drawbacks:
  - Switching from one ring to another one requires a context switch
    $\Longrightarrow$ each context switch consumes CPU time
  - If an application in the guest operating system requests the execution of a privileged instruction, the VMM provides a replacement function, which commands the execution via the kernel API of the host operating system
    $\Longrightarrow$ speed losses

Full Virtualization Examples

- Some virtualization solutions, which implement the VMM concept:
  - VMware Server, VMware Workstation and VMware Fusion
  - Microsoft Virtual PC (in the x86 version)
  - Parallels Desktop and Parallels Workstation
  - VirtualBox
  - Kernel-based Virtual Machine (KVM)
  - Mac-on-Linux (MoL)

Kernel-based Virtual Machine (KVM)

- KVM is integrated as a module directly in the Linux kernel
  - KVM core module: `kvm.ko`
  - Mardware-specific modules: `kvm-intel.ko` and `kvm-amd.ko`



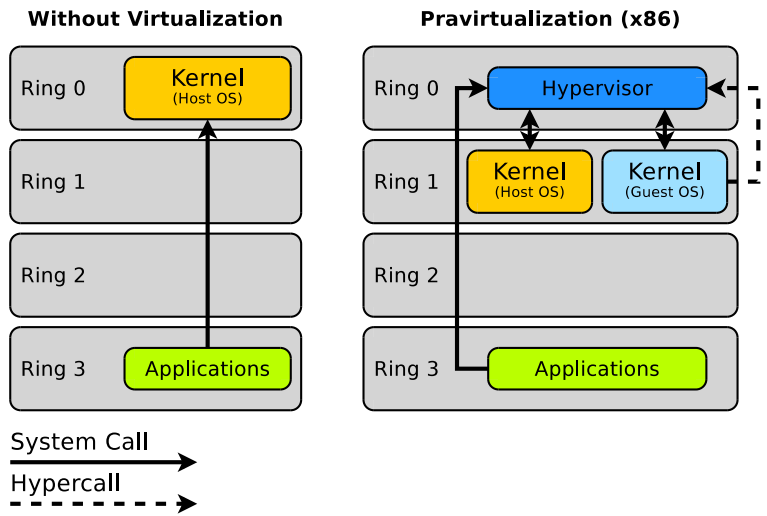
- After loading the modules, the kernel itself operates as a hypervisor
- KVM can only operate with CPUs, which implement hardware virtualization
  - Thus, KVM requires less source code as e.g. Xen
- Besides the kernel modules, KVM contains the emulator QEMU
  - KVM does not provide virtual hardware. This is provided by QEMU
    - CPU virtualization provides the CPU (Intel VT or AMD-V)
    - Main memory and storage is virtualized by KVM
    - I/O is virtualized by a dedicated QEMU process per guest

Paravirtualization (1/4)

- No hardware is virtualized or emulated
  - Does not provide an emulated hardware layer to the guest operating systems, but only an application interface
- Guest operating systems use an abstract management layer ($\Longrightarrow$ **hypervisor**) to access the physical resources
  - Hypervisor is a **meta operation system**, which is reduced to a minimum
    - The hypervisor distributes hardware resources among the guest systems, the same way, an operating system would distribute hardware resources among running processes
    - The hypervisor is a **Type-1 hypervisor** and runs *bare metal*
  - A meta operation system allows the independent operation of different applications and operating systems on a single CPU
- The hypervisor runs in located in the privileged ring 0
  - The host operating system is relocated to the less privileged ring 1
    - A host operating system is required because of the device drivers

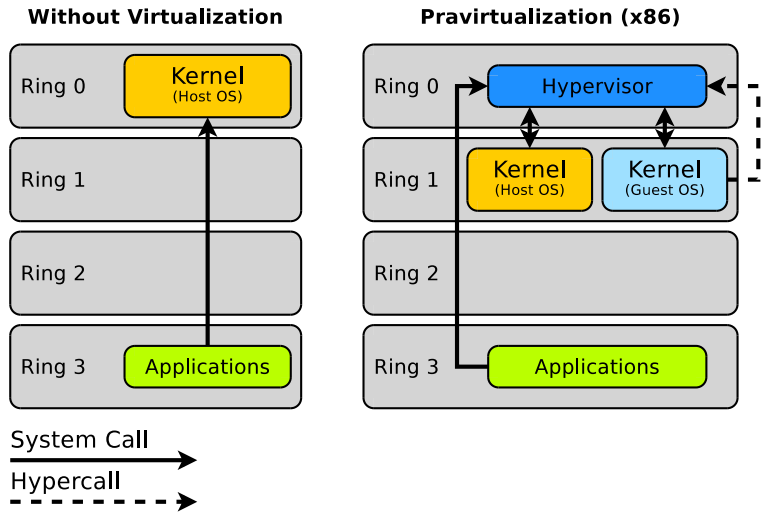
Paravirtualization (2/4)

- The host operating system is relocated from ring 0 to ring 1
  - Therefore, the kernel can not execute privileged instructions
  - Solution: The hypervisor provides **hypercalls**
- Hypercalls are similar to system calls
  - The interrupt numbers are different
  - If an application requests the execution of a system call, a replacement function in the hypervisor is called
  - The hypervisor orders the execution of the system call via the kernel API of the operating system



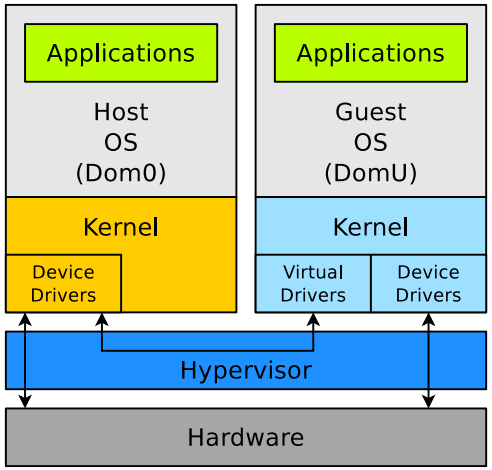
Paravirtualization (3/4)

- Kernels of guest operating systems need to be modified in a way that any system call for direct access to hardware is replaced by the corresponding hypercall
- Catching and verifying system calls by the hypervisor causes just little performance loss



- Examples: Xen, Citrix Xenserver, Virtual Iron, VMware ESX Server

Paravirtualization (4/4)

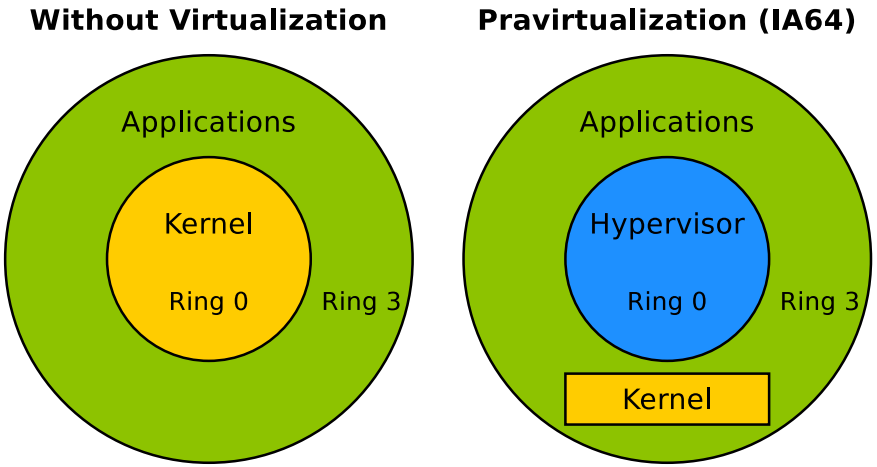


- VMs are called **unprivileged domain** (DomU)
- The hypervisor replaces the host operating system
  - But the developers can not develop all drivers from scratch and maintain them
    - Therefore, the hypervisor launches an (Linux) instance with its drivers and borrows them
    - This instance is called Domain0 (Dom0)

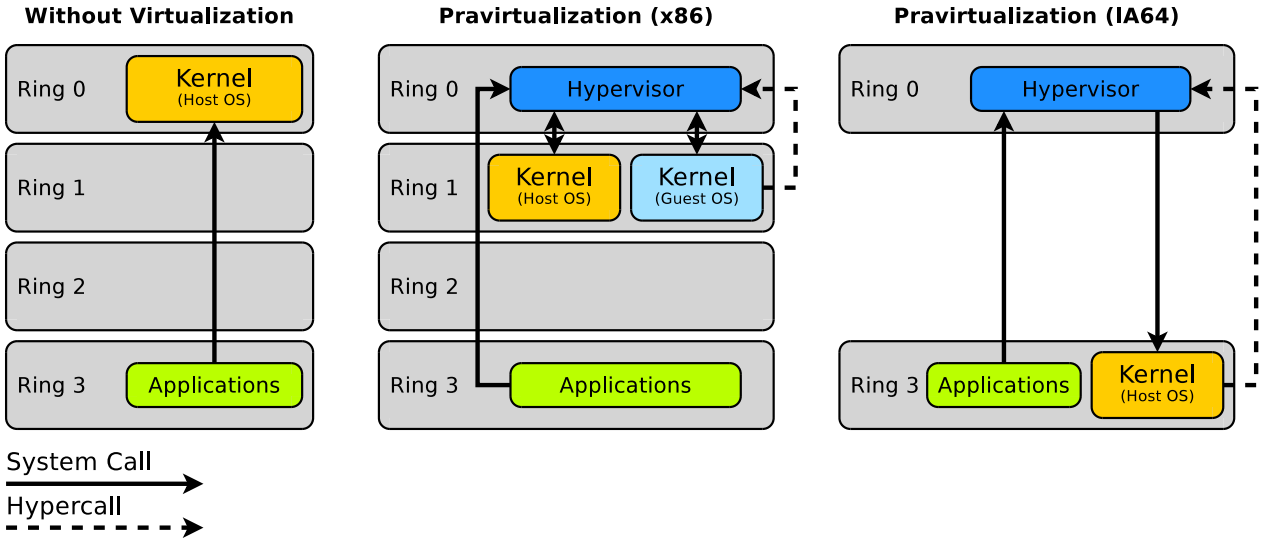
- Drawbacks:
  - Kernels of guest operating systems must be modified (adapted) for operation in the paravirtualized context
  - Rights holders of proprietary operating systems often reject an adjustment because of strategic reasons
    $\Longrightarrow$ Often works only with open source operating systems
- Advantage:
  - Better performance compared with VMM implementations

Problem: x86-64 Architecture

- The x86-64 architecture (e.g. IA64) does not implement ring 1 and 2



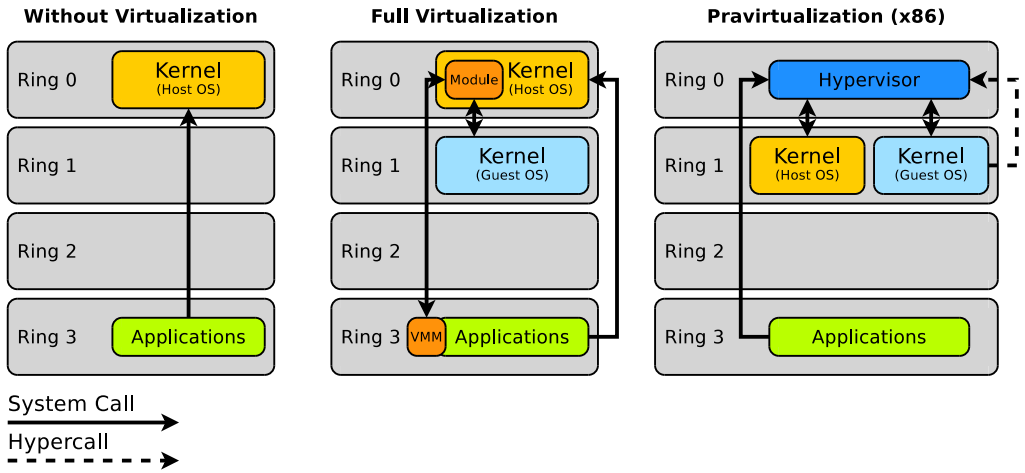
- In the x86-32 architecture, the hypervisor is located in ring 0
- In the x86-64 architecture, the operating system kernel is relocated to ring 3, where the applications are located



- Locating hardware drivers and applications in the same ring tends to be insecure

Summary: Virtualization vs. Pravirtualization

- **Paravirtualization** requires modified guest systems
  - Type-1 hypervisor runs *bare metal* (= replaces the host operating system)
  - Hypervisor runs in ring 0 and has full access to the hardware
  - Examples: VMware ESX(i), Xen, Microsoft Hyper-V
- **Full virtualization** supports unmodified guest systems
  - VMM (Type-2 hypervisor) runs *hosted* as an application in the host operating system
  - VMM runs in ring 3 at the level of the applications
  - Examples: VMware Workstation, KVM, Oracle VirtualBox, Parallels



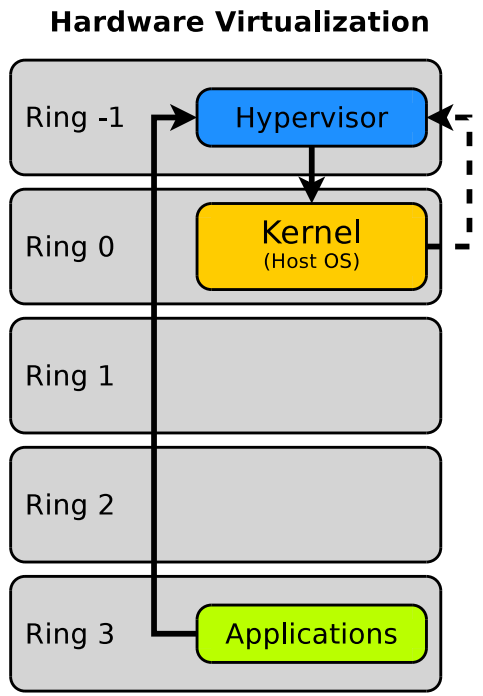
Hardware Virtualization (1/2)

- Current CPUs from Intel and AMD contain virtualization extensions for hardware virtualization
  - Advantage: Unmodified operating systems can be used as guest systems
  - The solutions from Intel and AMD are similar but incompatible
- Since 2006, AMD64 CPUs contain the Secure Virtual Machine (**SVM**) instruction set
  - The solution is called **AMD-V** and was previously called **Pacifica**
- The solution from Intel is called **VT-x** for IA32 CPUs and **VT-i** for Itanium CPUs
  - The solution of Intel was previously called **Vanderpool**

- Since Xen version 3, the software supports hardware virtualization
- Windows Server since Version 2008 (Hyper-V) uses hardware virtualization
- VirtualBox supports hardware virtualization
- KVM can only operate with CPUs, which implement hardware virtualization

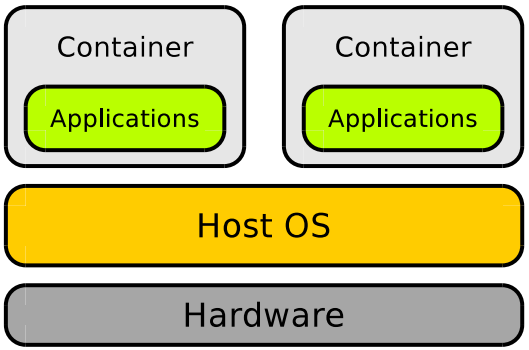
Hardware Virtualization (2/2)

- The hardware virtualization implementation contains a modification of the privilege levels
- A new ring ($\Longrightarrow$ ring -1) for the hypervisor is added
  - The hypervisor or VMM runs in ring -1 and at any time has the full control over the CPU and the resources, because with ring -1 an increased privilege level is implemented compared with ring 0
- VMs, executed inside ring 0 are called HVM
  - HVM = Hardware Virtual Machine
- Advantages:
  - Guest operating systems do not need to be modified (adapted)
    - Even proprietary operating systems (e.g. Windows) can be used as guest systems
  - In contrast to paravirtualization (IA64), the kernel is not executed in the privilege level of the applications



Operating System-level Virtualization / Containers (1/2)

- Under a single kernel, multiple identical, isolated system environments are executed
  - No additional operating system is started
    - An isolated runtime environment is created
  - All running applications use the same kernel
    - This kind of virtualization is called **Containers** in SUN/Oracle Solaris
    - This kind of virtualization is called **Jails** in BSD



- Applications only see applications from the same virtual environment
- One advantage is the low overhead, because the kernel manages the hardware as usual

- Drawback: All virtual environments use the same kernel
  - Only independent instances of the same operating system are started
  - It is impossible to start different operating systems at the same time

Operating System-level Virtualization / Containers (2/2)

- This type of virtualization is used to execute applications in isolated environments with high security
- Especially Internet service providers, which offer (virtual) root servers, or web services on multi-core processor architectures, use this type of virtualization
  - Little performance loss, high security level
- Examples:
  - SUN/Oracle Solaris (2005)
  - OpenVZ for Linux (2005)
  - Linux-VServer (2001)
  - FreeBSD Jails(1998)
  - Parallels Virtuozzo (2001, commercial version of OpenVZ)
  - FreeVPS
  - Docker (2013)
  - chroot (1982)

Docker

- Container are standard sized boxes to transport goods
- Docker offers a way to
  - package,
  - distribute and
  - run software.
- Containers can run on:
  - Linux (origin)
  - Mac (Beta, Yosemite 10.10)
  - Windows (Beta, Window 10 64 bit)
  - AWS
  - Azure

Docker Overview

Source: http://pointful.github.io/docker-intro/

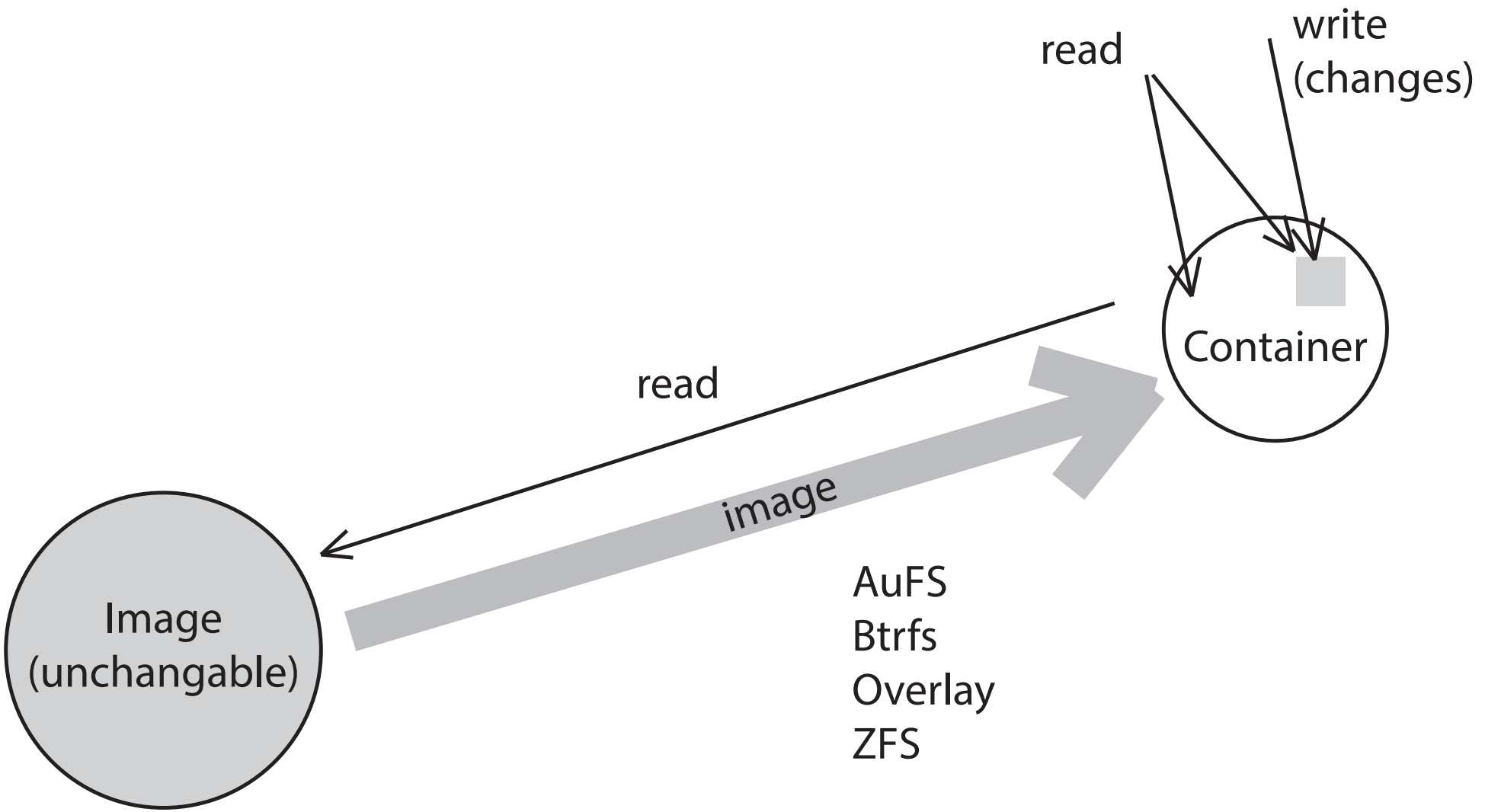
WHY

- Run everywhere
  - Regardless of kernel version (2.6.32+)
  - Regardless of host distro
  - Physical or virtual, cloud or not
  - Container and host architecture must match
- Run anything
  - If it can run on the host, it can run in the container
  - i.e. if it can run on a Linux kernel, it can run

WHAT

- High Level – It's a lightweight VM
  - Own process space
  - Own network interface
  - Can run stuff as root
  - Can have its own `/sbin/init` (different from host)
  - „machine container“
- Low Level – It's chroot on steroids
  - Can also not have its own `/sbin/init`
  - Container=isolated processes
  - Share kernel with host
  - No device emulation (neither HVM nor PV) from host)
  - „application container“

Docker Image Handling



Docker CLI (1/2)

List all local images:

```
>> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
```

Run nginx

```
>> docker run -d --name nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
... Pull complete ...
```

Images again:

```
>> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              0d409d33b27e        3 weeks ago         182.8 MB
```

List all processes:

```
>> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
                    PORTS               NAMES
02d5c39c3ba0        nginx               "nginx -g 'daemon off"   2 minutes ago       Up 2
    minutes         80/tcp, 443/tcp     nginx
```

Docker CLI (2/2)

Some options:

```
-d, --detach
-P, --publish-all
-p, --publish=[]
-i, --interactive
-t, --tty
```

Publish:

```
>> docker run -d -P --name nginx nginx
7cafcfa878a4761c458f1be9cc7d4f3481a7be8ecd83570f833b2a260287d81a
```

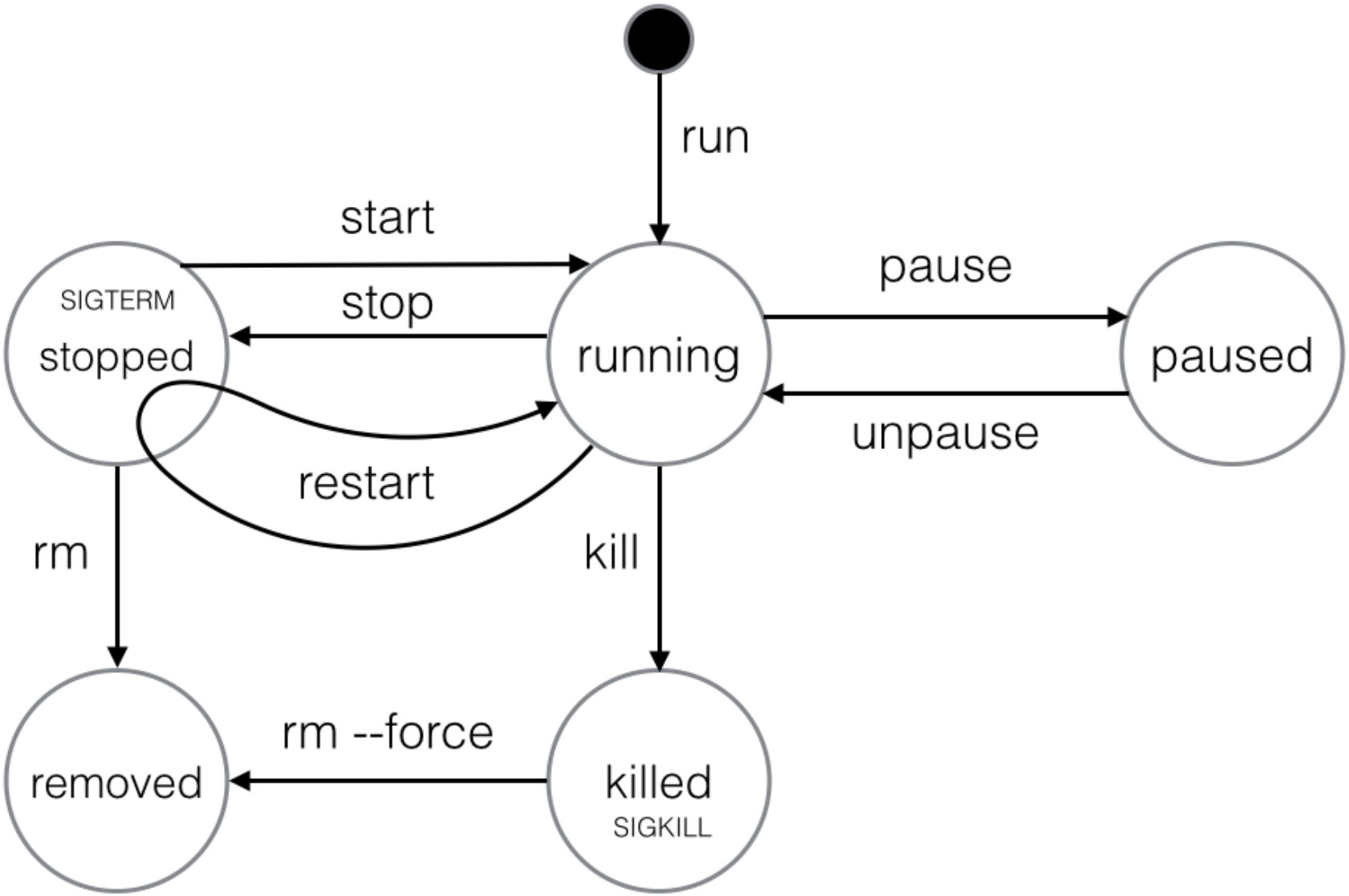
or

```
>> docker run -d -p 80:80 --name nginx nginx
3ee4629dd3233e617dc84343ae89543f919289cc927b643ac233574aa6b01e21
```

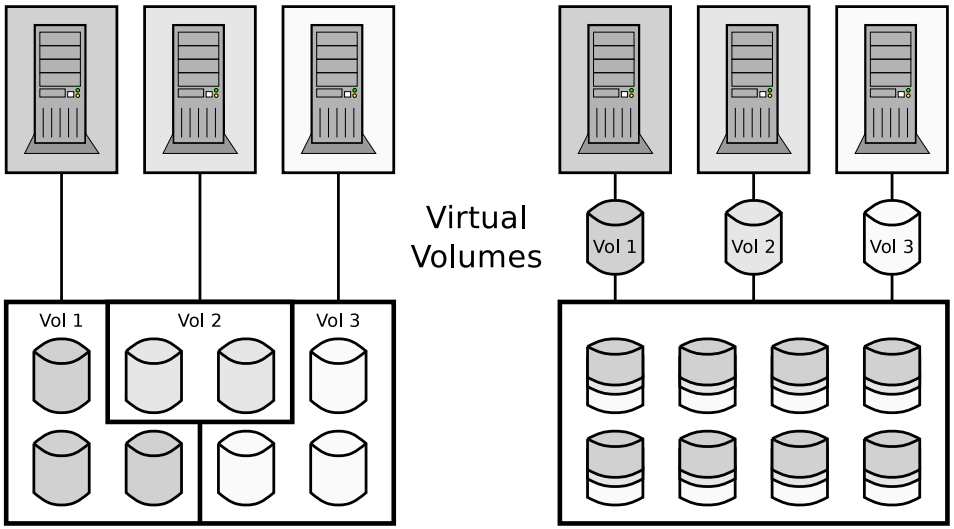
Processes again:

```
>> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS              PORTS                                NAMES
436552b60677        nginx               "nginx -g 'daemon off"   Less than a second ago
     Up 2 seconds        0.0.0.0:80->80/tcp, 443/tcp          nginx
```

Docker Container Lifecycle



Storage Virtualization



- Storage is provided to users in form of virtual drives (*volumes*)
- Logical storage is separated from physical storage

- Advantages:
  - Users are independent from the physical limits of drives
  - Reorganizing/expanding the physical storage does not disturb the users
  - Redundancy is provided transparently in the background
  - Better degree of utilization, because the physical storage can be split among the users in a more efficient way
- Drawback: Professional solutions are expensive
- Some Providers: EMC, HP, IBM, LSI and SUN/Oracle

Network Virtualization via Virtual Local Area Networks

- Distributed devices can be combined via VLAN in a single virtual (logical) network
  - VLANs separate physical networks into logical subnets (overlay networks)
    - VLAN-capable Switches do not forward packets of one VLAN into other VLANs
    - A VLAN is a network, over existing networks, which is isolated to the outside
  - Devices and services, which belong together, can be consolidated in separate VLANs
    - Advantage: Other networks are not influenced
      $\Longrightarrow$ Better security level

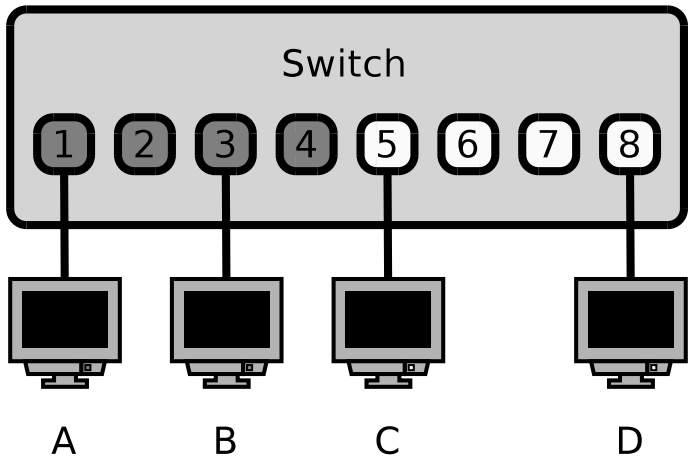
Helpful sources

- Benjamin Benz, Lars Reimann. *Netze schützen mit VLANs*. 11.9.2006
  `http://www.heise.de/netze/artikel/VLAN-Virtuelles-LAN-221621.html`
- Stephan Mayer, Ernst Ahlers. *Netzsegmentierung per VLAN*. c't 24/2010. S.176-179

VLAN Types

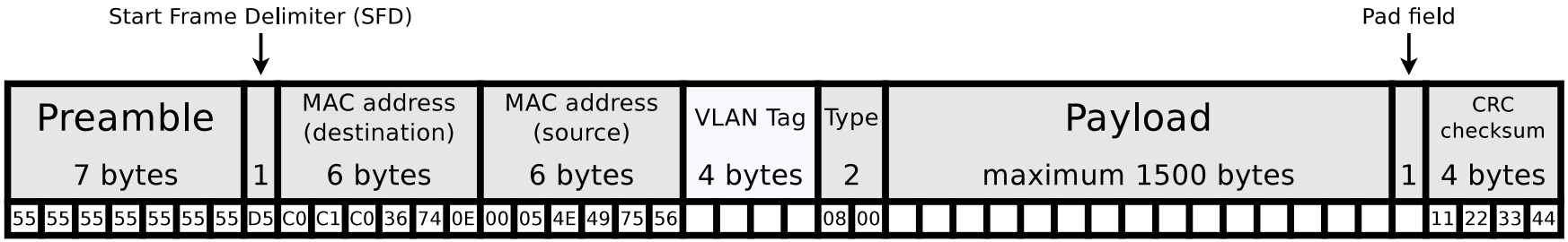
1. Oldest Standard: **Static VLAN**
   - The ports of a Switch are assigned to logical switches
   - Each port is permanently assigned to a VLAN or it connects different VLANs
   - Difficult to automate

Only node A and B, as well as node C and D can commuicate with each other, even though they are all connected with the same Switch

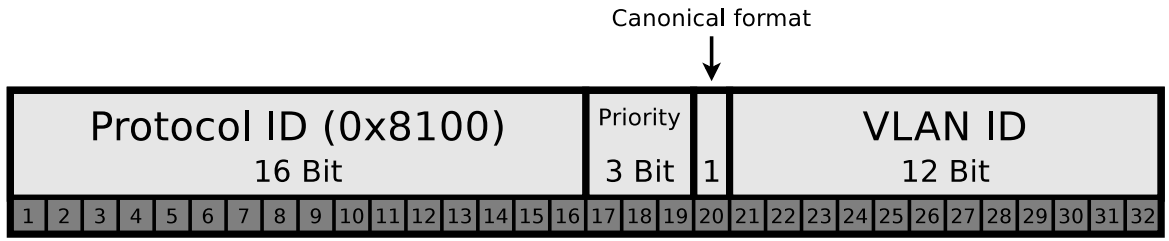


2. Latest: **Packet-based, dynamic VLAN** according to IEEE 802.1Q
   - Network layer packets contain a special VLAN *tag*
   - Dynamic VLANs can be created, changed and removed purely via software, using scripts

Ethernet Frame with VLAN Tag according to IEEE 802.1Q



- VLAN tag length: 32 bits



- Protocol ID (16 bits) is always set to 0x8100 for IPv4
- 3 bits are used to store the priority (QoS) value
  - 0 represents the lowest and 7 the highest priority
  - With this priority value, certain data (e.g. VoIP) can be prioritized
- Canonical format (1 bit) $\Longrightarrow$ most significant bit of the MAC addresses
  - 0 = Ethernet, 1 = Token Ring
- 12 bits contain the ID of the VLAN to which the packet in the frame belongs to

Examples of Useful Application Areas for VLANs

- **Telekom Entertain**
  - DSL connection with telephone line and IPTV ($\Longrightarrow$ *Triple Play*)
  - Uses 2 VLANs to transmit the IPTV traffic with a higher priority
    - „Normal“ internet via PPPoE via VLAN ID 7
    - IPTV without dialing via VLAN ID 8
- **Eucalyptus**
  - Private cloud infrastructure service (IaaS)
  - Each virtual machine (instance) is assigned to a security group
    - Each security group has its own firewall rules
  - Eucalyptus can be create for each security group a separate VLAN
    - Isolation of the traffic of instances according to the security groups
- **Data centers** or **home office**
  - Separation of the traffic according to economic aspects
  - Objective: Protect against operator errors and defective software
    - One VLAN for a „production network“ with the critical services
    - An additional VLANs for experiments, project work or children's games

Reasons for using Virtualization (1/2)

- Better hardware utilization
  - Server consolidation: Merge (virtual) servers on fewer physical servers
    - Reduction of costs for hardware, electric energy consumption, cooling, floor space, administration, etc.
- Simplified administration
  - Number of physical servers is reduced
  - Sophisticated management tools exist
  - VMs can be relocated during operation (live migration)
- Simplified deployment
  - New infrastructures and servers can be started manually or automatically within minutes

Reasons for using Virtualization (2/2)

- Maximum flexibility
  - VMs can be easily duplicated and backed up
  - Snapshots of the current state of a VM can be created and restored
- Increased security level
  - VMs are isolated against other VMs and the host system
  - Business critical applications can be encapsulated in a VM and run in a secure environment this way
  - Failure of a VM has no influence to other VMs or the host
- Optimization of software tests and software development
  - Simultaneous operation of multiple operating systems
  - Test environments can be set up quickly
- Support for old applications
  - Legacy operating systems or legacy applications, for which hardware is hard to obtain, can be reanimated

Drawbacks and Limitations of Virtualization

- Performance loss
  - Modern virtualization technologies are so much sophisticated, that the estimated performance loss is about 5-10%
  - Since modern computer systems provide hardware multi-core CPUs with support for hardware virtualization (Intel VT/VT-x and AMD-V), the performance loss plays an increasingly subordinate role
- Not all hardware can be addressed or emulated
  - Hardware dongles are not always compatible
  - Hardware-accelerated graphic output is hard to realize
- During failure of one host, multiple virtual servers fail
  - Concepts to handle failures and redundant installations are required
- Virtualization is complex
  - Additional know-how is required

Virtualization in Cloud Computing

- **Application virtualization** (JVM) in platforms like GAE
- **Partitioning** is only used in fields like cloud gaming because of the height acquisition costs
- **Full virtualization** is mainly used by cloud service providers, which use KVM to implement their service offerings
  - KVM is supported by multiple private cloud solutions
- Xen (**Paravirtualization**) is the basis of the AWS
  - Many private cloud solutions support Xen
- **Operating system-level virtualization** can help in cloud environments to utilize the hardware in a more efficient way
- **Storage virtualization** allows in cloud data centers to consolidate storage and utilize it in a more efficient way
- Some public and private cloud services use **VLANs** to separate the network communication of instances from the production network of the physical infrastructure

Virtualization – Conclusion

- Virtualization offers a big potential for savings, but also creates new attack points, especially at hypervisor level
  - VMware distributes the slim virtualization solution ESXi 3.5
    - A virtualization kernel, which has a size of only 32 MB, and a minimal operating system, running bare metal on the hardware
- Virtualization gets more important in the next years because...
  - its better energy efficiency, while the electricity costs rise
  - multi-core processors are standard today
  - hardware virtualization is available in almost all modern CPUs
- Hardware emulation $\neq$ Virtualization
  - Emulation tries to duplicate a different hardware architecture