10th Slide Set
Operating Systems

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`

Learning Objectives of this Slide Set

- At the end of this slide set You know/understand. . .
  - **virtualization** and its benefits, drawbacks and limitations
  - the difference between **hardware emulation** and virtualization
  - the architectures, benefits and drawbacks of different **virtualization concepts**

Exercise sheet 10 repeats the contents of this slide set which are relevant for these learning objectives

Virtualization – Fundamentals and General Idea

- 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 from the operating system instances are transparently intercepted 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

Virtualization Concepts

History of Virtualization

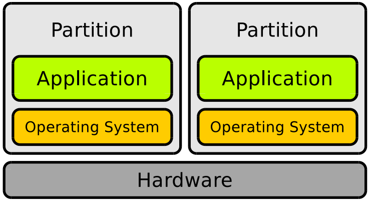
- 1970/71: IBM introduced the Virtual Machine Facility/370 (VM/370)
- On this platform, multi-user operation is implemented by using multiple single-user mode instances, which are executed in virtual machines
- Each VM is a complete duplicate of the underlying physical hardware

Creasy RJ. **The origin of the VM/370 time-sharing system**. IBM Journal of Research and Development (1981), No. 5, P.483–490

- Different virtualization concepts exist:
  - Partitioning
  - Hardware emulation
  - Application virtualization
  - Full virtualization (Virtual Machine Monitor = Type-2 Hypervisor)
  - Paravirtualization (Type-1 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

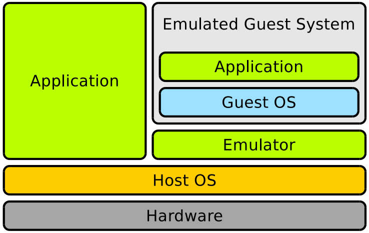
Image (Watson stage replica in Jeopardy! contest, Mountain View, California): Atomic Taco. `flickr.com` (CC-BY-SA-2.0)
Image (Interns demonstrating Watson capabilities in Jeopardy! exhibition match): Rosemaryetoufee. Wikimedia (CC-BY-SA-4.0)

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 simulates the **entire hardware** of a computer system, for running an **unmodified operating system** designed for a **different hardware architecture** (CPU)
  - Exception: Wine (https://www.winehq.org)
    - 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: QEMU, Bochs, PearPC, Wabi, DOSBox, M.A.M.E.



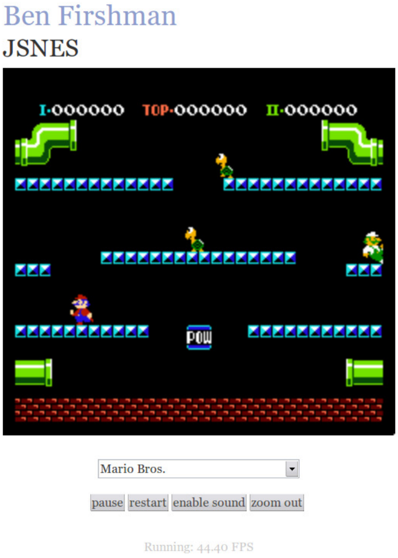
Selection of Emulators

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
Basilisk 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-License	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 until 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
- `https://jsnes.org`
- `github.com/bfirsh/jsnes`
- Free Software (GPLv3)

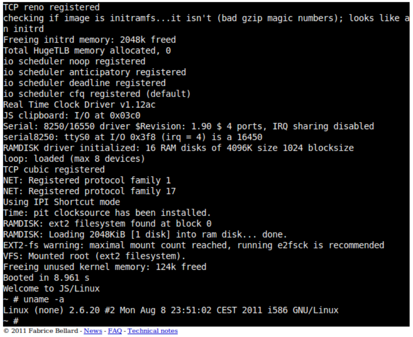


Latest Development: Browser emulates PC – jslinux

`https://www.wired.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**.

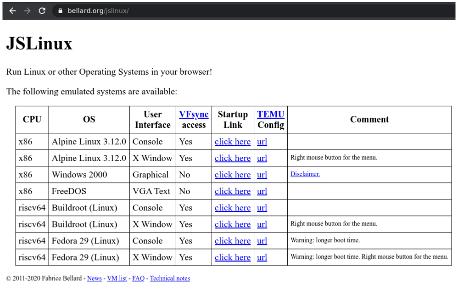


```
TCP reno registered
checking if image is initramfs...it isn't (bad gzip magic numbers); looks like a
n initrd
Freeing initrd memory: 2048k freed
Total HugeTLB memory allocated, 0
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
Real Time Clock Driver v1.12ac
JS clipboard: I/O at 0x03c0
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing disabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16450
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI Shortcut mode
Time: pit clocksource has been installed.
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 2048KiB [1 disk] into ram disk... done.
EXT2-fs warning: maximal mount count reached, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 124k freed
Booted in 8.961 s
Welcome to JS/Linux
~ # uname -a
Linux (none) 2.6.20 #2 Mon Aug 8 23:51:02 CEST 2011 i586 GNU/Linux
~ #
```

© 2011 Fabrice Bellard - News - FAQ - Technical notes

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.


bellard.org/jslinux/

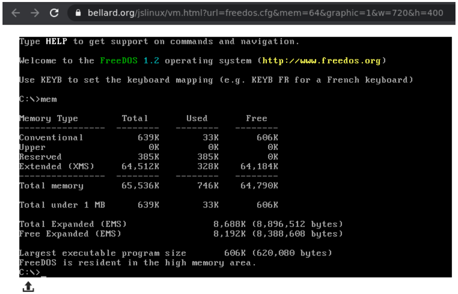
JSLinux

Run Linux or other Operating Systems in your browser!

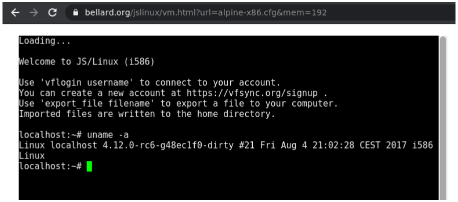
The following emulated systems are available:

CPU	OS	User Interface	VFsync access	Startup Link	TEMU Config	Comment
x86	Alpine Linux 3.12.0	Console	Yes	click here	url	
x86	Alpine Linux 3.12.0	X Window	Yes	click here	url	Right mouse button for the menu.
x86	Windows 2000	Graphical	No	click here	url	Disclaimer.
x86	FreeDOS	VGA Text	No	click here	url	
riscv64	Buildroot (Linux)	Console	Yes	click here	url	
riscv64	Buildroot (Linux)	X Window	Yes	click here	url	Right mouse button for the menu.
riscv64	Fedora 29 (Linux)	Console	Yes	click here	url	Warning: longer boot time.
riscv64	Fedora 29 (Linux)	X Window	Yes	click here	url	Warning: longer boot time. Right mouse button for the menu.

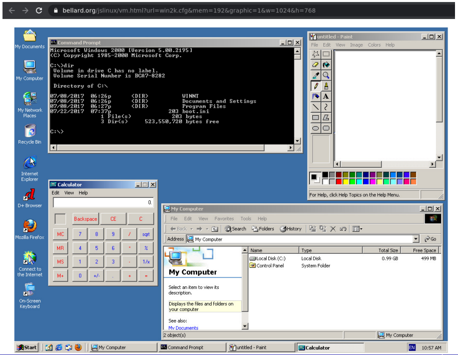
© 2011-2020 Fabrice Bellard - News - VM list - FAQ - Technical notes


bellard.org/jslinux/vm.html?url=freedos.cfg&mem=64&graphic=1&w=720&h=400

```
Type HELP to get support on commands and navigation.

Welcome to the FreeDOS 1.2 operating system (http://www.freedos.org)

Use KEYB to set the keyboard mapping (e.g. KEYB FR for a French keyboard)

C:\>mem

Memory Type         Total       Used       Free
----------------  --------   --------   --------
Conventional          639K        33K       606K
Upper                   0K         0K         0K
Reserved              385K       385K         0K
Extended (XMS)     64,512K       328K    64,184K
----------------  --------   --------   --------
Total memory       65,536K       746K    64,790K

Total under 1 MB      639K        33K       606K

Total Expanded (EMS)                8,688K (8,896,512 bytes)
Free Expanded (EMS)                 8,192K (8,388,608 bytes)

Largest executable program size       606K (620,080 bytes)
FreeDOS is resident in the high memory area.
C:\>_
```


bellard.org/jslinux/vm.html?url=alpine-x86.cfg&mem=192

```
Loading...

Welcome to JS/Linux (i586)

Use 'vflogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export_file filename' to export a file to your computer.
Imported files are written to the home directory.

localhost:~# uname -a
Linux localhost 4.12.0-rc6-g48ec1f0-dirty #21 Fri Aug 4 21:02:28 CEST 2017 i586
Linux
localhost:~# 
```


bellard.org/jslinux/vm.html?url=win2k.cfg&mem=192&graphic=1&w=1024&h=768

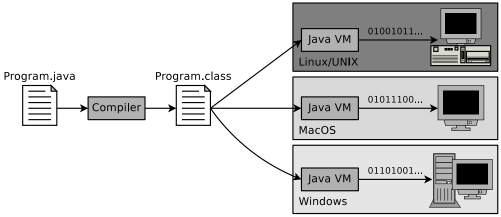
```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is BCA7-8282

 Directory of C:\

07/08/2017  06:26p      <DIR>          WINNT
07/08/2017  06:26p      <DIR>          Documents and Settings
07/08/2017  06:27p      <DIR>          Program Files
07/22/2017  07:37p                 203 boot.ini
               1 File(s)            203 bytes
               3 Dir(s)     523,550,720 bytes free

C:\>
```

- Since 2011, the author of JSLinux has added a lot of new features

Image top right: FreeDOS 1.2 (x86)
Image bottom left: Alpine Linux 3.12.0 (x86)
Image bottom right: Windows 2000 (x86)

Application Virtualization

- Applications are executed inside a virtual environment, which uses local resources and provides all the components the application needs
  - The VM is 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



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

- Advantage: Platform independence
- Drawback: Less performance, compared to native program execution

VMware ThinApp

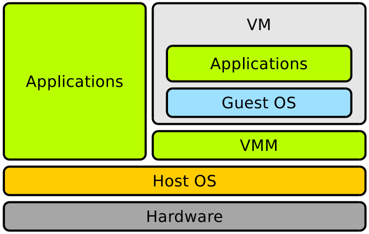
http://www.vmware.com/products/thinapp/

- 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

The boundaries between Application Virtualization and Operating System-level Virtualization / Containers (see slide 26) are not sharp

Full Virtualization (1/3)

- Full virtualization software solutions offer 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 devices, 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 by multiple operating systems
  - Example: Network adapters
  - The emulation of popular hardware avoids driver issues



Virtualization Basics of the x86 Architecture

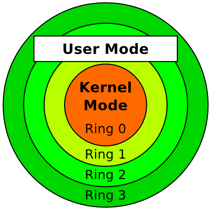
- x86-compatible CPUs implement 4 privilege levels
  - Objective: Improve stability and security
  - Each process is assigned to a ring permanently

- Implementation: The register CPL (Current Privilege Level) stores the current privilege level
- Source: `http://css.csail.mit.edu/6.858/2012/readings/i386.pdf`

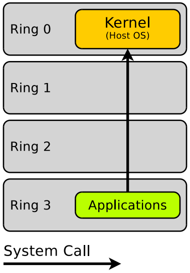
- In ring 0 (= **kernel mode**) runs the kernel
  - Processes here have full hardware access
- In ring 3 (= **user mode**) run the applications
  - Processes are in Protected Mode ($\Longrightarrow$ slide set 5)

Unmodified modern operating systems only use 2 privilege levels (rings) $\Longrightarrow$ Rings 0 and 3

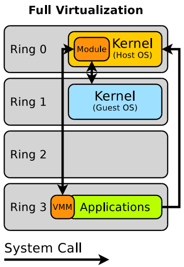
- If a user-mode process must carry out a higher privileged task (e.g. access hardware), it can tell this the kernel via a **system call** ($\Longrightarrow$ slide set 7)
  - The user-mode process generates an exception, which is intercepted in ring 0 and handled there



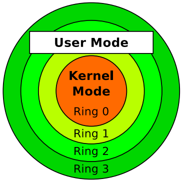
Without Virtualization

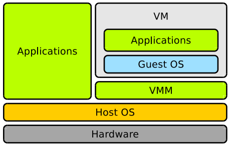


Full Virtualization (2/3)



- 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 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)

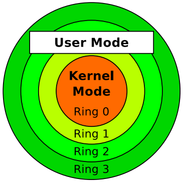
- Advantages:
  - Few modifications in the host and guest operating systems are required
  - Access to the main resources is just forwarded (passed through)
    $\Longrightarrow$ guest operating systems run almost with native performance
  - Each guest operating system has its own kernel
    $\Longrightarrow$ high degree of flexibility
- Drawbacks:
  - Switching from one ring to another one requires a process/context switch
    $\Longrightarrow$ each process/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

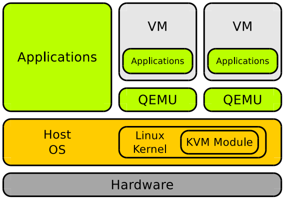
Full Virtualization Examples: Some virtualization solutions, which implement the VMM concept

- VirtualBox
- VMware Server, VMware Workstation and VMware Fusion
- Parallels Desktop and Parallels Workstation
- Kernel-based Virtual Machine (KVM) $\Longrightarrow$ The architecture of KVM is different... (see next slide)

Kernel-based Virtual Machine (KVM)

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





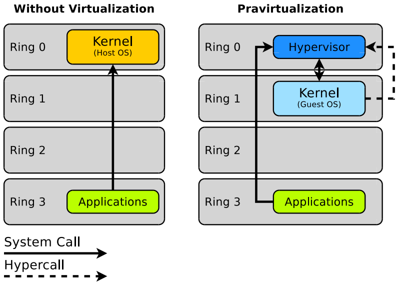
- After loading the modules, the kernel itself operates as a hypervisor
- KVM can only operate with CPUs, which implement hardware-assisted virtualization
- Besides the kernel modules, KVM contains the emulator QEMU
  - KVM does not provide virtual hardware. This is provided by QEMU
    - CPU virtualization 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 (1/2)

- 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 the privileged ring 0
  - The guest operating systems are relocated to the less privileged ring 1
    - The hypervisor must include all required device drivers or as alternative, a guest operating system instance that includes the required device drivers is mandatory

Paravirtualization (2/2)

- Guest operating system kernels cannot execute privileged instructions
  - Solution: The hypervisor provides **hypercalls**



Hypercalls are similar to system calls

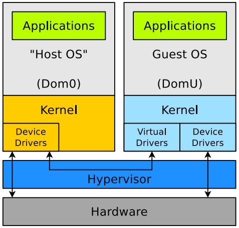
- The interrupt numbers are different
- If an application requests the execution of a system call, a replacement function in the hypervisor is called
- The hypervisor orders the execution of the system call via the kernel API of the operating system

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

Paravirtualization Examples: Some virtualization solutions, which implement the paravirtualization concept

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

Xen

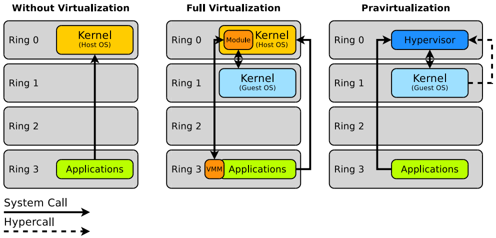


- VMs are called **unprivileged domain** (DomU)
- The hypervisor replaces the host operating system
  - But the developers can not develop all drivers from scratch and maintain them
    - Therefore, the hypervisor launches an (Linux) instance with its drivers and borrows them
    - This instance is called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

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



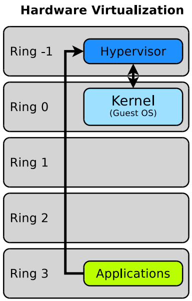
Hardware-assisted Virtualization (1/2)

- Modern CPUs from Intel and AMD contain virtualization extensions for Hardware-assisted 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 2008 (Hyper-V) uses hardware virtualization
- VirtualBox supports hardware virtualization
- KVM can only operate with CPUs, which implement hardware virtualization

Hardware-assisted Virtualization (2/2)

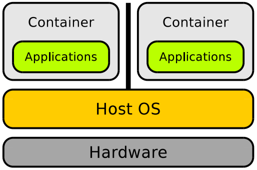
- The hardware virtualization implementation contains a modification of the privilege levels
- A new ring ($\Longrightarrow$ ring -1, called *root mode*) for the hypervisor is added
  - The hypervisor or VMM runs in ring -1 and at any time has the full control over the CPU and the other resources, because 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

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

- Under a single kernel, multiple identical, isolated system environments are executed
  - No additional operating system is started
    - Isolated runtime environments are 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

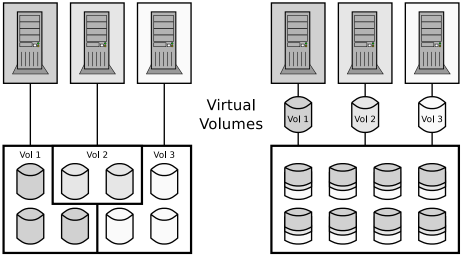


- Applications only see applications from the same virtual environment
- One advantage is the low overhead, because the kernel manages the hardware as usual
- Drawback: All virtual environments use the same kernel
  - Only independent instances of the same operating system are started
  - It is impossible 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)

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 a 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

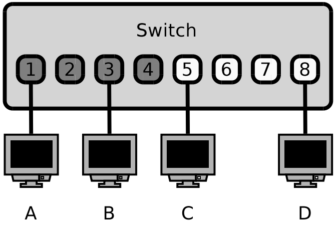
Helpful sources

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

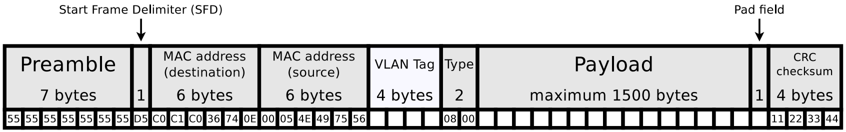
1. Oldest Standard: **Static VLAN**
   - The ports of a Switch are assigned to logical switches
   - Each port is permanently assigned to a VLAN or it connects different VLANs
   - Difficult to automate

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

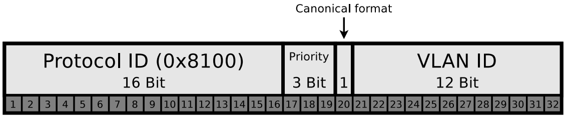


2. Latest: **Packet-based, dynamic VLAN** according to IEEE 802.1Q
   - Network layer packets contain a special VLAN *tag*
   - Dynamic VLANs can be created, changed and removed purely via software, using scripts

Ethernet Frame with VLAN Tag according to IEEE 802.1Q


Start Frame Delimiter (SFD)
Pad field
Preamble 7 bytes
1
MAC address (destination) 6 bytes
MAC address (source) 6 bytes
VLAN Tag 4 bytes
Type 2
Payload maximum 1500 bytes
1
CRC checksum 4 bytes
55 55 55 55 55 55 55 D5 C0 C1 C0 36 74 0E 00 05 4E 49 75 56 08 00 11 22 33 44

- VLAN tag length: 32 bits



Structure of the VLAN Tag

- Protocol ID (16 bits) is always set to 0x8100 for IPv4
- 3 bits are used to store the priority (QoS) value. 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** (`https://www.eucalyptus.cloud`)
  - 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

- Better hardware utilization
  - Server consolidation: Merge (virtual) servers on fewer physical servers
    - Less costs for hardware, electric energy, cooling, floor space, etc.
- Simplified administration
  - Number of physical servers is reduced
  - Sophisticated management tools exist
  - VMs can be relocated, duplicated and backed up during operation
  - Snapshots of the current state of a VM can be created and restored
- Simplified deployment
  - Infrastructures and servers can be started (automatically!) within minutes
- Increased security level
  - VMs are isolated against other VMs and the host system
    - Critical applications can be encapsulated and run in a secure environment
  - Failure of a VM does not influence other VMs or the host
- Support for legacy operating systems or legacy applications
  - Independence from obsolete hardware (End-of-Life-Hardware)