



Module I-7410 Advanced Linux FS-11 Part1: Virtualization with KVM

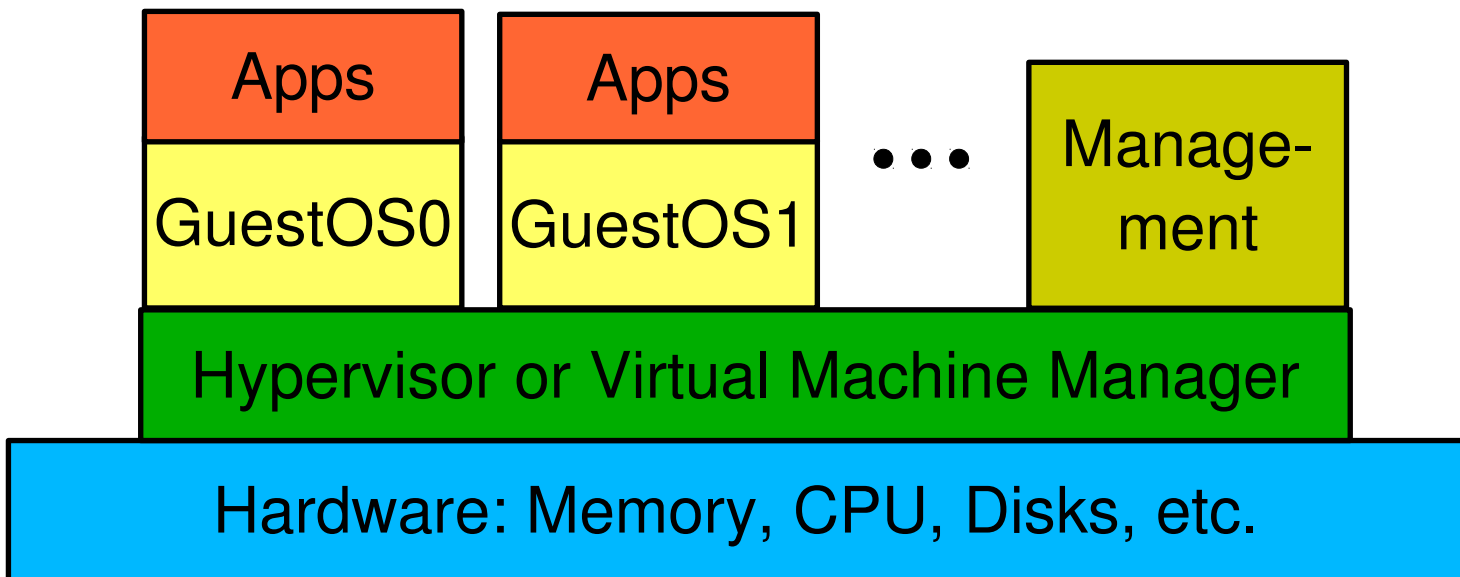
By Franz Meyer
Version 1.0
February 2011





Virtualization Architecture

- Full Virtualization
 - Uses a hypervisor to share the underlying hardware (bare metal hypervisor). It must run at the most privileged level (ring0)
 - The guest operating system need not be modified: It is not aware that the underlying hardware is virtualized. It runs at lower privilege level.
 - Expect significant performance degradation





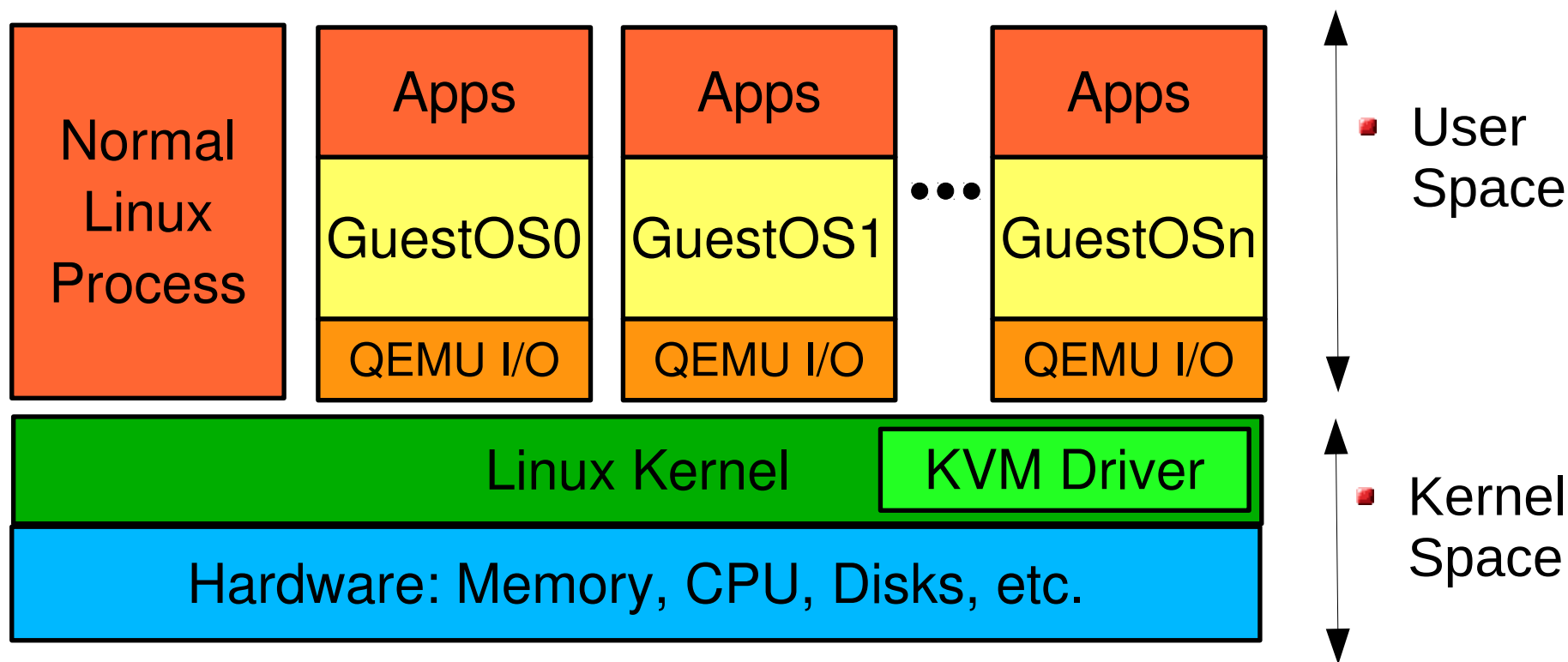
Hardware Assisted Virtualization

- Some Intel and AMD provide extensions to simplify CPU virtualization (Intel-VT-X, AMD-V).
- The principle:
 - A CPU can operate in host or guest mode.
 - In guest mode the hypervisor sees whether the guest operates in kernel or user space.
 - Privileged operations of the guest transfer control to the hypervisor of the host.
 - The virtualization support reduces the overhead for CPU virtualization of the machine.
 - The hypervisor provides virtual memory mapping between the memory of the physical host and the guest (shadow page tables).



KVM Architecture

- QEMU
 - Emulation for all unsupported devices (network, disk, display)
- KVM Driver
 - Support for HW-Virtualization (Intel-VT or AMD-V)





KVM Architecture

- QEM is a software that emulates a number of hardware platforms in software. This is time consuming and slows down virtualization.
- KVM is implemented as a kernel loadable module (kvm driver). It converts the host system in a bare metal hypervisor that supports QEMU for X-86 based kernels. KVM is sometimes called QEMU accelerator.
- A virtual machine in KVM is implemented as a regular Linux process, scheduled by the standard Linux scheduler.
- Device emulation is handled by QEMU that provides an emulated BIOS, PCI Bus, USB bus, IDE and SCSI disk controllers, network cards, etc.
- The KVM code has been incorporated into the Linux kernel since kernel 2.6.20 (2007). Thus, the KVM code improves with the kernel. Notice that XEN code is not part of the Linux kernel.



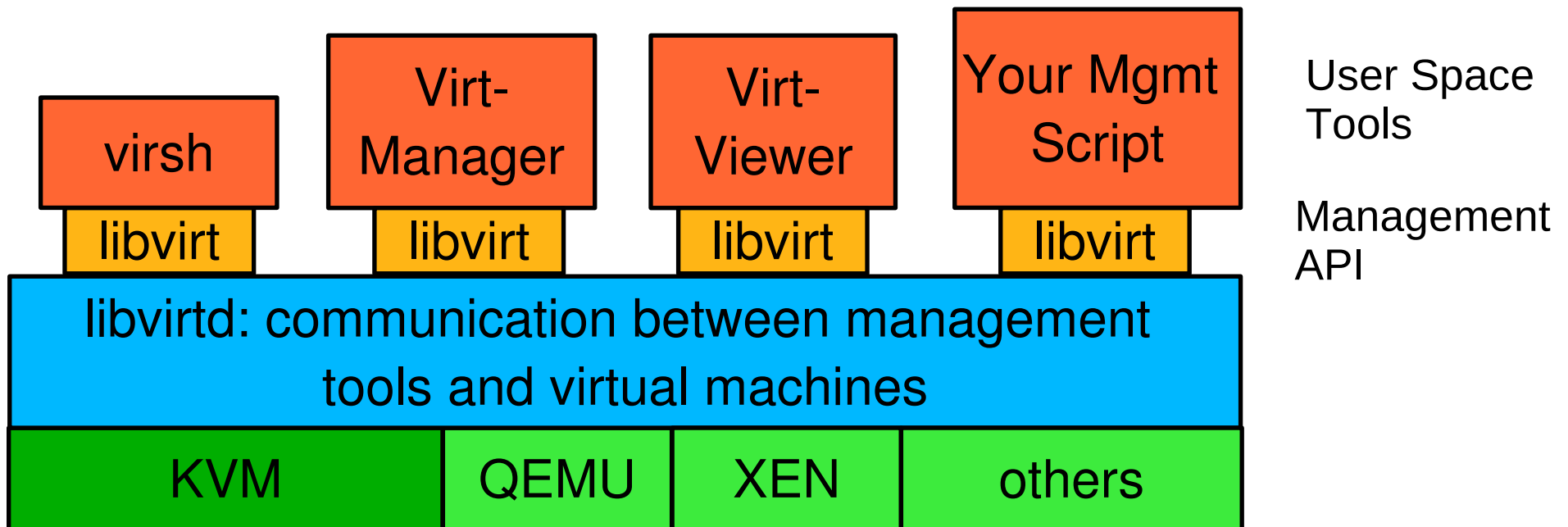
KVM Pros and Cons

- Pros
 - Reduces performance degradation by using HW-Virtualization
 - No customization needed on the guest OS
 - QEMU supports various devices (ex. VNC Server - Display)
- Cons
 - Needs a modified QEMU
 - Not yet as fast as para-virtualized systems
 - No installation-tool yet with debootstrap



KVM Components

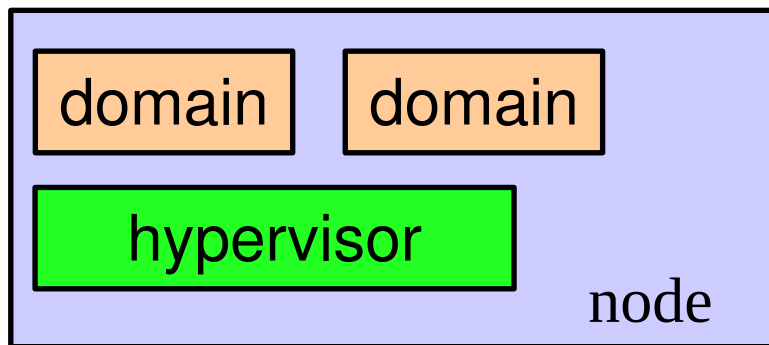
- The following picture shows the most important KVM components
- Libvirt unifies the management of many different virtualization systems (kvm, qemu, xen, virtualbox, etc, right now 9)





Libvirt - Terminology

- In libvirt the following terms are used (<http://www.libvirt.org>)
 - **Node:** is a single physical
 - **Hypervisor:** is a layer of software allowing to virtualize a node in a set of virtual machines with possibly different configurations than the node itself
 - **Domain** is an instance of an operating system (or subsystem in the case of container virtualization) running on a virtualized machine provided by the hypervisor





Libvirt Management - Tools

- Virsh (/usr/bin/virsh)
 - Command line tool for libvirt (shell like)
- Virt-installer (/usr/bin/virt-install, command line tool)
 - Creates guestOS .xml and image files based on input parameters
- Virt-cloner (/usr/bin/virt-clone, command line tool)
 - Clones existing virtual machines
- Virt-viewer (/usr/bin/virt-viewer)
 - VNC based viewer for virtual machines
 - Used by the other tools
- Virt-Manager (/usr/bin/virt-manager)
 - GUI-client to libvirt,
 - Can also connect on remote libvirt-daemons
- Read the manual pages for each of those tools



Libvirt (1)

- Objective of Libvirt: unifies the management of virtual machines
- Stores configuration in xml format
- Supports among other things
 - create/configure/destroy guests
 - create/configure/destroy storage pools
 - create/configure/destroy virtual networks (no bridges)
- Configuration of libvirt: directory **/etc/libvirt**
 - File: **libvirtd.conf** - libvirt daemon configuration file
 - File: **qemu.conf** - configuration file for the QEMU driver
 - Directory: **qemu** - configuration xml for kvm/qemu based clients
 - A .xml file for each virtual machine
 - Directory: **network** - .xml configuration files for virtual networks
 - Directory: **storage** - .xml configuration files for storage pools



Libvirt (2)

- Xml schemes are explained in detail here: <http://www.libvirt.org>
- Libvirt defines interfaces for programming languages like C, C#, Python, Perl, etc.
- The libvirtd can be configured to allow network access.



Bridge Management Tools (1)

- Allows you to create/configure virtual Ethernet bridges
- Installation (as root): **apt-get install bridge-utils**
- Adding physical interface **eth0** to a bridge called **br0**.
 - Modify file **/etc/network/interfaces**

static case **auto eth0**

```
iface eth0 inet manual
```

```
auto br0
```

```
iface br0 inet static
```

```
Address 192.168.0.100
```

```
Network 192.168.0.0
```

```
Netmask 255.255.255.0
```

```
Broadcast 192.168.0.255
```

```
Gateway 192.168.0.1
```

```
bridge_ports eth0
```

```
bridge_fd 9
```

```
bridge_hello 2
```

```
bridge_maxage 12
```

```
bridge_stp off
```

dhcp case

```
auto eth0
```

```
iface eth0 inet manual
```

```
auto br0
```

```
iface br0 inet dhcp
```

```
bridge_ports eth0
```

```
bridge_fd 9
```

```
bridge_hello 2
```

```
bridge_maxage 12
```

```
bridge_stp off
```



Bridge Management Tools (2)

- Bridge utilities
 - **brctl** - ethernet bridge administration
 - Syntax is: **brctl [command]**
 - Commands are
 - `brctl addbr <name>` creates a new instance of the ethernet bridge called <name>
 - `brctl delbr <name>` deletes the instance <name> of the bridge
 - `brctl show <brname>` will show some information on the bridge and its attached ports
 - More information: **\$ man brctl**
- Bridge utilities can be used for the GuestOS configuration in libvirt.



KVM Lab: Part1 (1)

- Objective
 - Learn how to plan, setup, and manage a small number of guest virtual machines using kvm virtualization
 - You will be familiar with the appropriate tools to do that
 - Libvirt based tools such as
 - virsh: the command line shell to libvirt
 - virt-install: the command line tool to create a guest
 - virt-clone: the command line tool to create a guest
 - virt-viewer: a GUI based tool for displaying the graphical console of a guest
 - Virt-manager: desktop GUI for managing virtual machines



KVM Lab: Part1 (2)

- Task-1: Prepare your system for kvm and libvirt
 - Step-1: Make sure that virtualization is enabled in the BIOS and that the CPUs of the Lab PC support hardware virtualization.
 - Step-2: Install kvm, libvirtd, virt-installer, virt-viewer, bridge-utils
 - Step-3: Reboot your system
- Task-2: Configure your network with the following requirements:
 - Configure a bridge called br0 including network interface eth0.
 - Configure a virtual network called **internal** using the subnet settings associated to your lab-machine.
- Task-3:
 - Step-1: Create a LVM volume group of 15 Gb that holds virtual machine images.
 - Step-2: Configure a storage pool to hold virtual machine images. Perhaps start with a LV and use the pool later.



KVM Lab Part1 (3)

- Task-4: Create a guest named **router** which connects to the bridged ethernet **br0** and to the virtual network **internal**.
 - Step-1: Create basic xml file with the **virt-install** utility and install the GuestOS from an installation CD.
 - Step-2: If the guest is started automatically stop it and add a second network card in the xml file.
 - Step-3: Start and configure the guestOS.
- Task-5: Similarly to task 4. Create a guest named **server** which only connects to the virtual network **internal**.
 - Step-1: Create guest by cloning the first guest (router)
 - Step-2: Remove the second network card from the configuration.
- Task-6: Check connectivity from a machine on the Netlab network segment to the **server** across the **router**.
- All tasks: document your work.



KVM Lab Part2 (4)

- Objectives
 - Learn manage guest on different nodes (physical machine)
 - You will be familiar with the techniques to access a common storage, to remotely manage a guest, and to migrate a guest from one node to another node.
- Remark: It was found that some task do not work properly in Ubuntu-10.4 but more successfully 10.10 or in Debian-squeeze.
- Task-1: Plan and setup a second node. Are there any restrictions concerning the network topology?
 - Step-1: Get a second disk and setup a Linux distribution and install the kvm software, perhaps clone your existing disk.
 - Step-2: Setup identical virtual networks and bridges.
 - Step-3: Achieve to share the storage for virtual machine images between all nodes, perhaps using NFS.



KVM Lab Part2 (5)

- Task-2: Install one or more guests in the common storage.
- Task-3: Manage those guests
 - Step-1: Find out how instances of virtlibd on different nodes communicate. Install the required software.
 - Step-2: Use the **virsh** utility to start a guest on a remote node. Then connect to that machine or use virt-viewer.
 - Step-3: Use the **virsh** utility to start a guest on the local node. Then migrate this guest to the remote node. Check that the migration has been successful. Then migrate the guest back to your local machine.
- Document your work.



Resources

- KVM - <https://help.ubuntu.com/community/KVM>
 - Installation
 - Networking
 - Guest Creation
 - Guest Management
- KVM - <http://wiki.debian.org/KVM>
- Libvirt - <http://www.libvirt.org>
- Virsh: Managing guests with virsh – <http://docs.redhat.com>