Introduction to virtualization and KVM

LINAGORA

Contents

Origins

Objectives : pros and cons

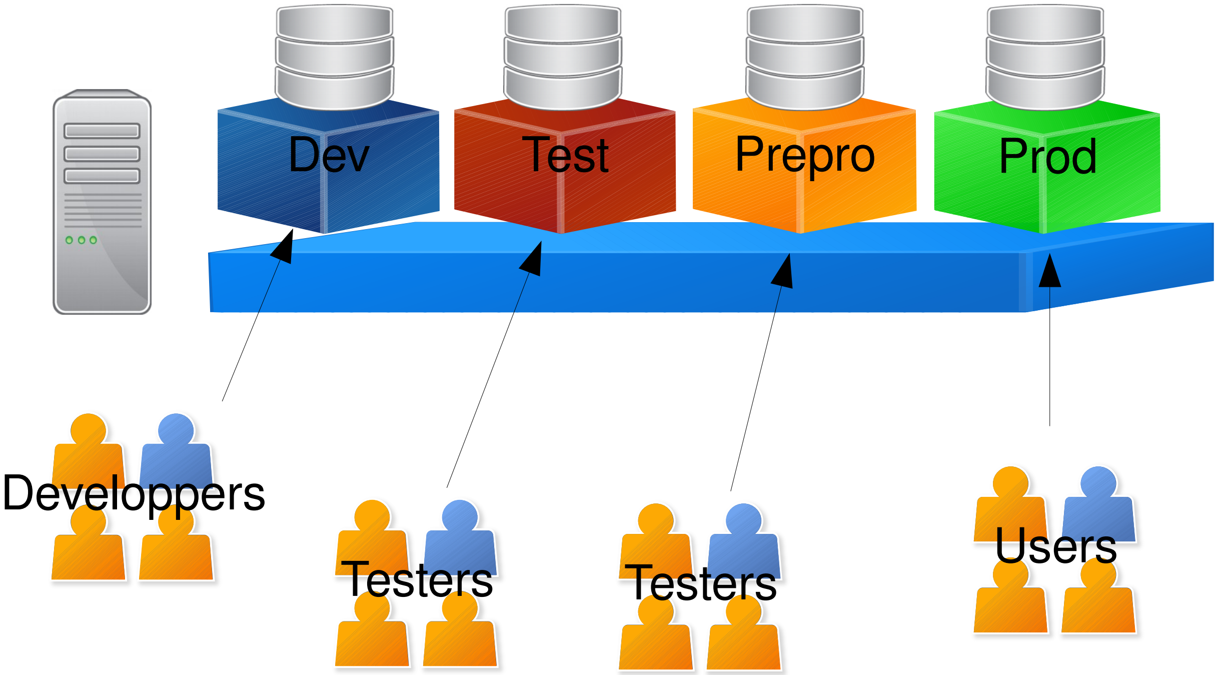
Technologies and examples

KVM

Origins

Mainframes

- In the 60’s up to 80’s
  - Mainframes
  - Only one (very expensive) computer, many environments
  - A solution :
    - environments are isolated
    - Their own adjustable resources



On Intel architectures

- In the 90’s and 2000’s
  - Virtualization on x86
  - Software emulation
  - Hardware virtualization (cpu instruction set)
  - Available on low cost hardware
  - A lot of free and non-free are available
    - Xen, KMV, Bochs, QEMU, Linux Vserver, VirtualBox
    - VirtualPC, VirtualServer, VMWare Server

Objectives

Objectives

- OS or application as a simple software
  - In a virtual environment (VE)
  - As many computers and many OS on a single computer
- Tests
  - Isolated VE
  - Migrate VE to more adaptated hardware
- Principles
  - A host OS
  - A virtualisation software : hypervisor
  - Other OS : guests

Cost reduction

- energy consumption
  - servers
  - air conditioning
  - Green factor
- hardware maintenance delays
  - Reduce maintenance procedures
  - ... the administration time consumption
- Reduce hardware compatibility management
  - few components expenditures
  - few suppliers to manage
  - Many versions of software can run longer
- Hardware used efficiently
  - high cost components (CPU, RAM,...) are used more or less full time
  - Consolidation
  - Less hardware
  - Less room space

Easier maintenance

- Reduce maintenance procedures
  - Install from predifined models
  - With automated IT infrastructucture integration (network, backup agent, supervision agent, ...)
- Central monitoring
  - A central supervision point
  - To decide migration action
  - To react
- Start/stop VM without restart physical server
  - Snapshots : quick restore

Disavantages

Limitations

- Compatibility
  - Software may not be compatible with virtualization
  - Then exceptions to manage
- Security
  - Hardware = single point of failure
  - Data protection with delegated data center
  - Server sharing
- VM out of control
  - To much VM where fewer are needed
  - Orphan VM

Technologies

Host OS-level

- Host OS isolates user spaces
  - One user space = one execution environment
  - No guest OS
  - Low overhead rate = high performances
  - Host OS dependant
- Examples
  - Chroot
  - Docker
  - Linux Vserver
  - LXC

Full virtualization

- Hypervisor first generation
  - Host OS with light kernel
  - Optimize transfers between guests OS and hardware
- Guest OS adaptation ?
  - No
  - If yes, paravirtualization (guests and host OS's cooperation)
- High performances because thin layer
- Examples
  - Xen
  - Vsphere
  - Hyper-V server

Full virtualization

- Hypervisor second generation
  - Close to emulation hardware components
  - Good isolation
  - Thicker layer between hardware and guests OS
  - Performances reduced
- Examples
  - VirtualBox
  - KVM
  - QEMU
  - Bochs

Kernel-based Virtual Machine (KVM)

Introduction

Preparation

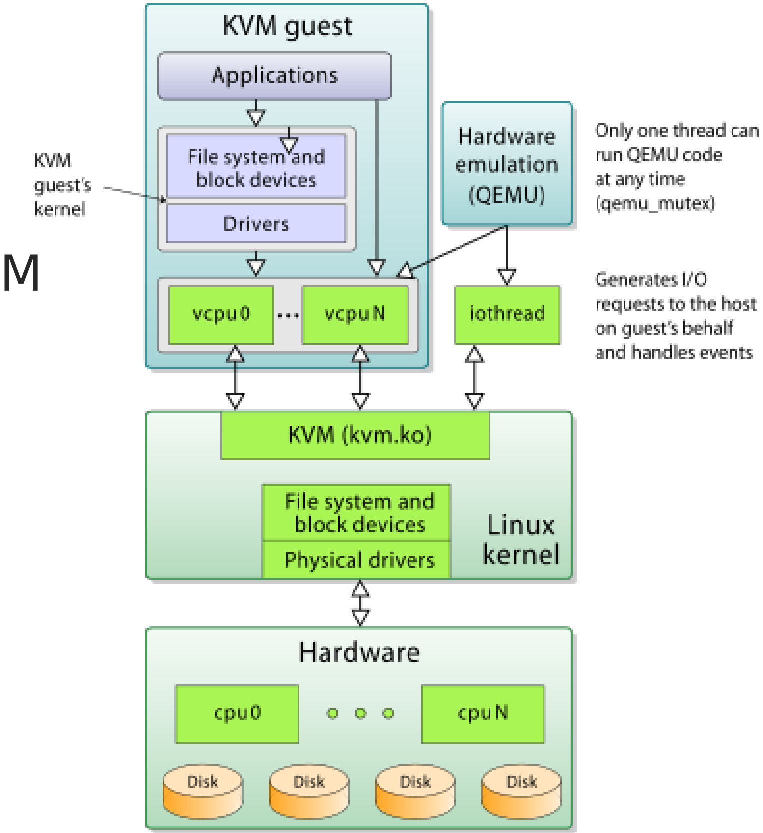
Guest creation

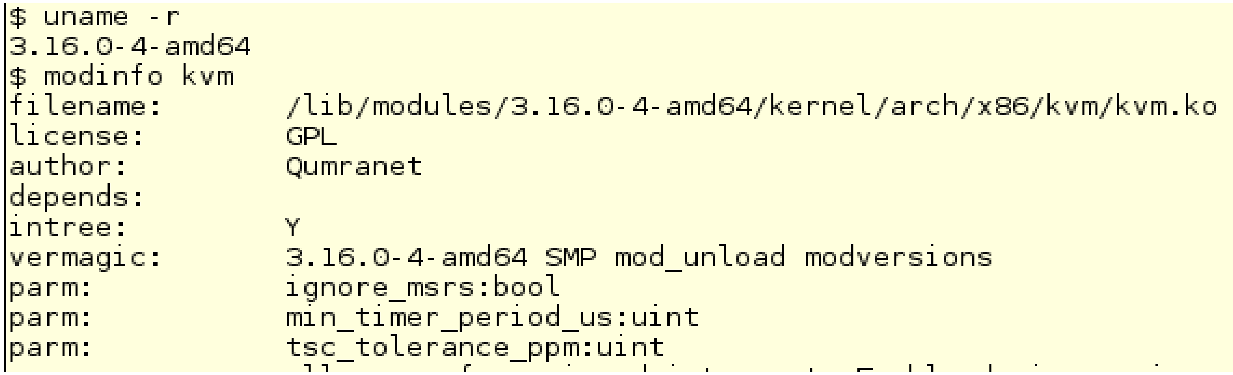
VMs management

Introduction

What is KVM ?

- Hypervisor in a kernel module
  - In Linux mainline since 2007
  - Also on FreeBSD
  - Architectures : x86, S/390, PowerPC, ARM
- Guests
  - Linux, BSD, Solaris
  - Windows
  - OS X
- I/O hardware emulation
  - Based on QEMU
  - In user- space

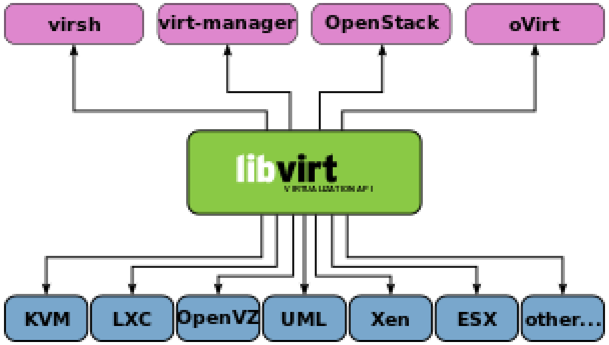

KVM guest
Applications
File system and block devices
Drivers
KVM guest's kernel
vcpu0
...
vcpuN
Hardware emulation (QEMU)
Only one thread can run QEMU code at any time (qemu_mutex)
iothread
Generates I/O requests to the host on guest's behalf and handles events
KVM (kvm.ko)
File system and block devices
Physical drivers
Linux kernel
Hardware
cpu0
cpuN
Disk
Disk
Disk
Disk



```
$ uname -r
3.16.0-4-amd64
$ modinfo kvm
filename:       /lib/modules/3.16.0-4-amd64/kernel/arch/x86/kvm/kvm.ko
license:        GPL
author:         Qumranet
depends:
intree:         Y
vermagic:       3.16.0-4-amd64 SMP mod_unload modversions
parm:           ignore_msrs:bool
parm:           min_timer_period_us:uint
parm:           tsc_tolerance_ppm:uint
```

Environment

- Open source
  - Since 2005 (Qumranet company)
  - Since 2008, RedHat
  - 2012 : v 1.2
- Processors with Virtualization Technology
  - Check BIOS
  - Check CPU (/proc/cpuinfo)
- Libvirt (*virsh, virt-manager*)
  - Independent and standard VM manager on a host
  - API to provision, create, modify, monitor, start, stop, migrate VM



Virtualized devices

- Virtualized and emulated
  - KVM emulates « full software » devices
  - Virtual CPUs
  - PCI bridges, USB controllers, IDE, AHCI controller, etc.
  - Graphics, sound, network (**e1000, rtl8139**)
- Para-virtualized devices
  - Faster I/O transfers, higher I/O throughput
  - Based on kernel modules : **virtio-***
  - Network, disks, SCSI, etc
- Physical devices
  - Direct access to certain hardware : passthrough
  - Depends on platforms

Storage

- Storage pools
  - Local SP : attached to the host server (local devices)
  - Network shared SP
- Emulated storage devices
  - Virtio-scsi : recommended
  - IDE, CDROM, USB mass storage, AHCI for SATA
- Host storage
  - Raw for I/O performances
  - QCOW2 for advanced features : snapshots, encryption, compression

Preparation

Host hardware

For about 10 VM :

- CPU
  - At least 4 cores
  - Virtualization technology
- Memory
  - At least 8 GB
- Disk drive
  - At least 100 GB
- *Overcommitting technology*
  - it is possible to allocate more virtualized CPU/RAM than physical
  - KVM distributes CPU/RAM to VM’s

Packages

- Qemu-kvm
  - Qemu binaries
  - User-level emulator
- Libvirt
  - GUI tool
  - Virtualization management tools
  - Start
- Guestfs
  - Tool for offline images management

Guest requisites

- According to VM guest goals, choose :
  - number of CPU
  - RAM amount
  - Storage volume/technology and select host disk
  - Networking
    - Bridge (check bridge is set on host)
    - NAT
    - fixed or DHCP
  - OS installation method : medium, NFS, HTTP, copy from pattern, unattended

Guest creation

Creation

- Online command **virt-install**

  -
    ```
    virt-install \
    --network bridge:br0 \
    --name rh7dev \
    --ram=1024 \
    --vcpus=1 \
    --disk path=/opt/vm-images/rh7dev.img,size=10 \
    --cdrom /opt/iso/rhel7.3.iso
    ```
  - Other interesting options :

    --location : URL of the distribution tree installation

    --os-type : guest operating system

    --arch : if non-native architecture

    The disk is named **vda**.

Cloning

- Cloning is another way to create a guest
  - Stop all I/O on the VM to be cloned :

    ```
    virsh suspend rhel7dev
    ```

  - Run the cloning command :

    ```
    virt-clone \
              --connect qemu:///system \
              --original rhel7dev \
              --name rhel7dev-clone \
              --file /opt/vm-images/rhel7dev-clone.img
    ```

  - Resume original VM :

    ```
    virtsh resume rhel7dev
    ```

  - Start the cloned VM

    ```
    virsh start rhel7dev
    ```

VM management

Current tasks

- List VMs
  - `virtsh list –all`
- Show VM information
  - `virtsh dominfo rhel7dev`
- Show VCPU/Memory usage
  - `virt-top`
- Show VM disk partitions
  - `virt-df rhel7dev`
- Start VM
  - `virsh start rhel7dev`
- Stop VM (shutdown the OS)
  - `virsh shutdown rhel7dev`

Other tasks

- Access to VM console
  - **`virsh shutdown rhel7dev`**

    first add « **`console=tty0 console=ttyS0,115200`** » to kernel boot line in grub configuration of VM

- Attach/detach storage device
  -
    ```
    virsh attach-disk rhel7dev /dev/sdb vdb \
        --driver qemu --mode shareable
    ```

    `vdb :` device name mapped in th VM
  - **`virsh detach-disk rhel7dev vdb`**

Changing VM memory dynamically

- Change dynamically memory
  - View current’s memory settings

    ```
    virsh dominfo rhel7dev | grep memory
    Max memory :  1048576 kB
    Used memory : 1048576 kB
    ```

  - Set to 512 MB (value in kB)

    ```
    virsh setmem rhel7dev 524288

    virsh dominfo rhel7dev | grep memory
    Max memory :  1048576 kB
    Used memory : 524288 kB
    ```

Changing VM offline

- Increase memory
  - Stop VM

    ```
    virsh shutdown rhel7dev
    ```

  - Edit VM’s configuration file

    ```
    virsh edit rhel7dev
    <memory unit='KiB'>2097152</memory>
    ```

  - Restart the VM with the new parameter

    ```
    virsh create /etc/libvirt/qemu/rhel7dev.xml
    ```

  - Check memory settings

    ```
    virsh dominfo vm1 | grep memory
    Max memory:      2097152 kB
    Used memory:     524288 kB
    ```

Change VPCUs

- Stop the VM

```
virsh shutdown rhel7dev
```

- Modify the CPU in the configuration file

```
virsh edit rhel7dev
<vcpu>1</vcpu>
```

- Restart VM

```
virsh create /etc/libvirt/qemu/rhel7dev.xml
```

Add disk

- Create a 10 GB file

```
dd if=/dev/zero of=/opt/vm-images/rhel7dev-hd2.img
bs=1M count=10240
```

- Shutdown the VM

```
virsh shutdown rhel7dev
```

- Add a disk stanza in the VM configuration file

```
<disk type='file' device='disk'>
  <driver name='qemu' type='raw' cache='none’ io='threads'/>
  <source file='/opt/vm-images/rhel7dev-hd2.img'/>
  <target dev='vdb' bus='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06'
  function='0x0'/>
</disk>
```

- Restart

```
virsh create /etc/libvirt/qemu/rhel7dev.xml
```

Deleting VMs

- Shutdown the VM

```
virsh shutdown rhel7dev
```

- If the VM is not responding or fails to shutdown, shut it down forcefully

```
virsh destroy rhel7dev
```

- Undefine the VMs configuration

```
virsh undefine rhel7dev
```

- Remove the VM’s image files

```
rm /opt/vm-images/rhel7dev*.img
```

Improve I/O performances

- Use VirtIO drivers (disk, network)

```
<disk type='file' device='disk'>
   <source file='/opt/vm-images/disk1.img'/>
   <target dev='vda'  bus='virtio'/>
</disk>
```

- Assign device disk instead of « file » disk

```
<source file=’/dev/vgvm/lvdd1’/>
```

Thanks for your attention

LINAGORA – headquarters

80, rue Roque de Fillol
92800 PUTEAUX
FRANCE

Phone : +33 (0)1 46 96 63 63
Info : info@linagora.com
Web : www.linagora.com

@linagora

facebook.com/Linagora/