Heavy and Light Virtualization

Guillaume Urvoy-Keller
UCA/I3S

01/07/2022

Part I: Introduction

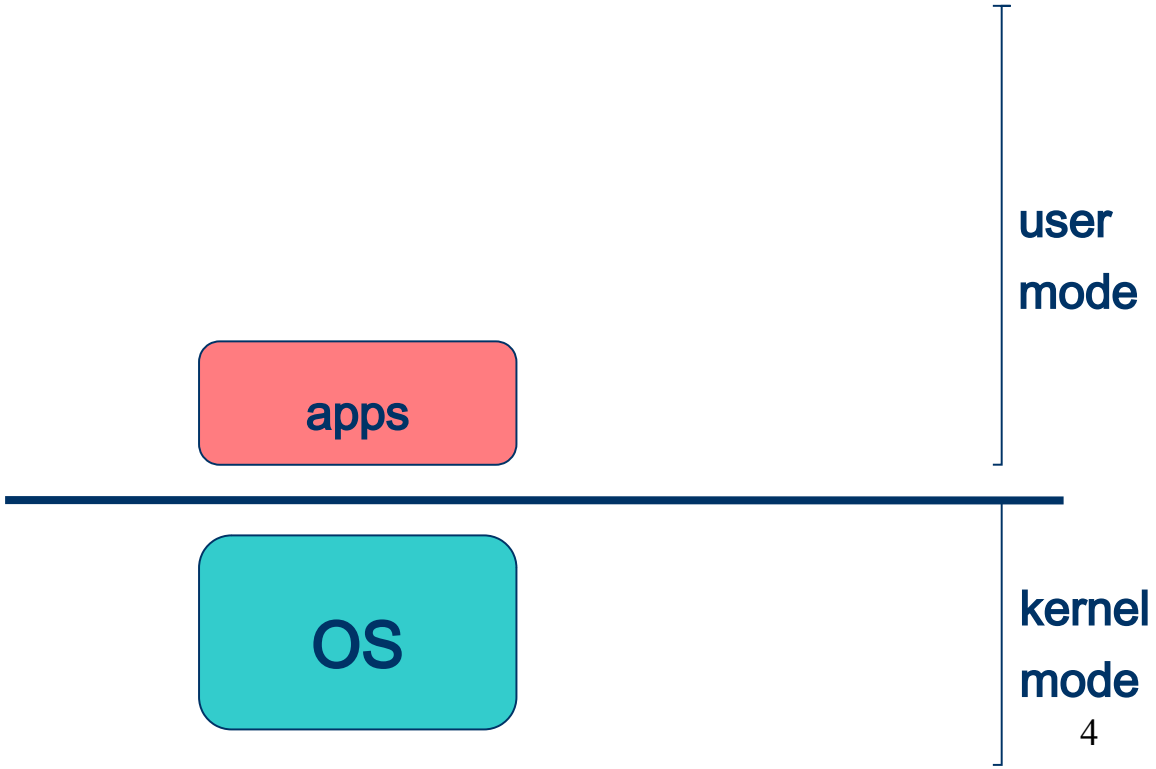
Introduction

- Set of techniques to enable several OS to run simultaneously on a physical machine/server (≠ **multi-boot)**

- **Virtualization entails a specific layer called a hypervisor, a.k.a, virtual machine monitor (VMM)**
  - **Mediator between physical resources and the OSes that are virtualized**

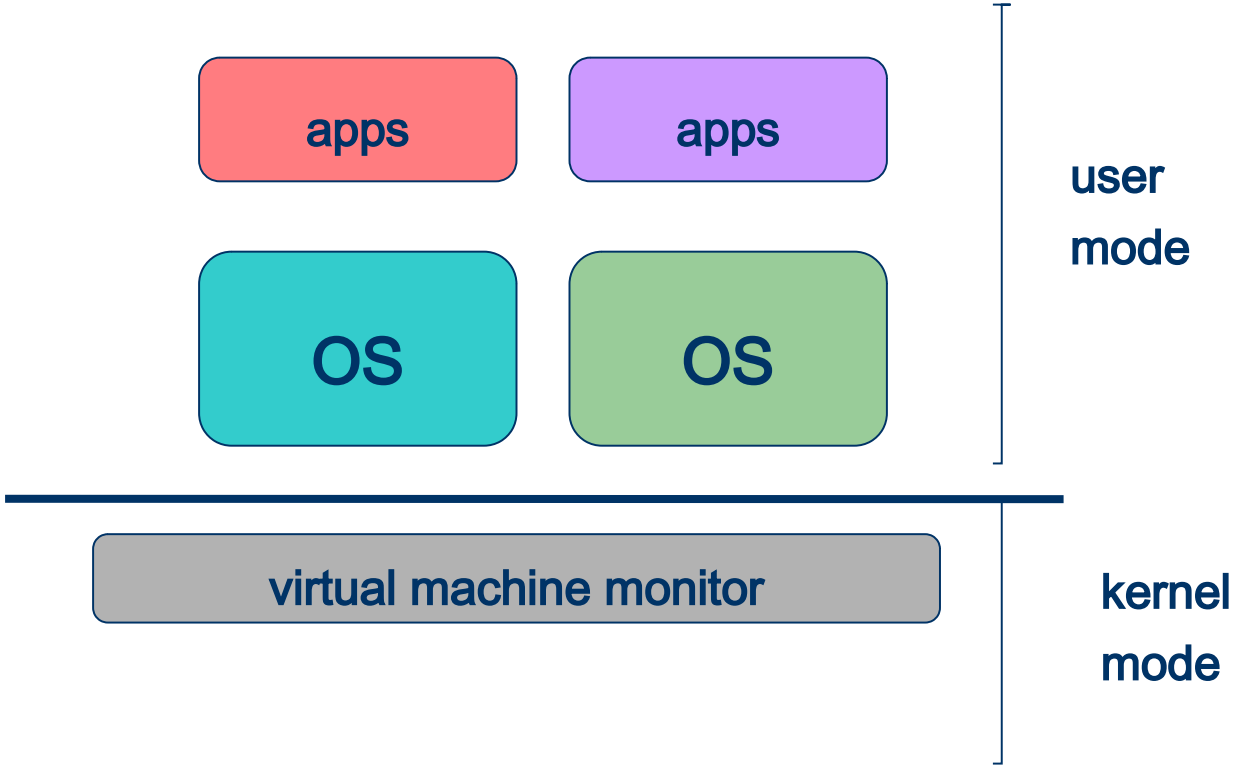
Virtualization = unpriviledging an OS

- Processors features two modes : user and kernel



Unpriviledging an OS

OS in user mode is called Guest OS



Hypervisors Zoo

- For servers :
  - VmWare Vsphere ~ 60% market share
  - Microsoft Hyper-V ~ 20%
  - Citrix XenServer ~ 4% - Amazon Web Services
  - QEMU/KVM – default openstack deployment
- For clients
  - Oracle Virtualbox
  - Vmware player
- Source : http://www.nasdaq.com/article/growing-competition-for-vmware-in-virtualization-market-cm316783

Why virtualizing?

- In the 90s, cost of servers decreased gradually
- Software editors (Microsoft, distribution Linux) advocate one application/service per server →
  - One DNS server
  - One mail server
  - One NFS server
- Each server with specific OS version and libraries
- Servers isolation

Why virtualizing

- Net result:
  - A host of servers in servers rooms (data centers)
  - 80% have an average utilization below 10%
  - Maintenance/operational costs increase with the number of servers
  - Server rooms are not infinitely extensible → lack of place
  - Non negligible electricity/air conditioning cost

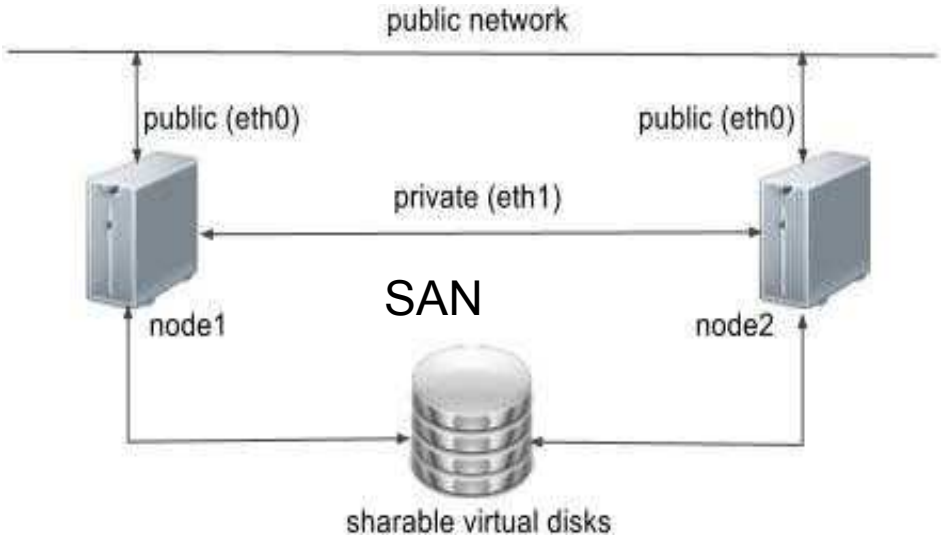
Why virtualizing

- Servers are less expensive but also more powerful
  - 64 bits multi-core with tons of RAMs are the norm
  - One server in 2009 is roughly estimated to be one order of magnitude more powerful than a server in 2004
- Replacing servers on a one to one basis is not an option any more
- Still, one wants to ensure service isolation

Advantages of virtualization

- Consider a company that has virtualized its IT
- Typical set-up for an SME:
  - Two high-end servers, a SAN (Storage Area Network) to share/consolidate storage between the servers
  - Licensing cost, e.g., Vmware
  - Training cost for IT service

Source : http://oracle-base.com

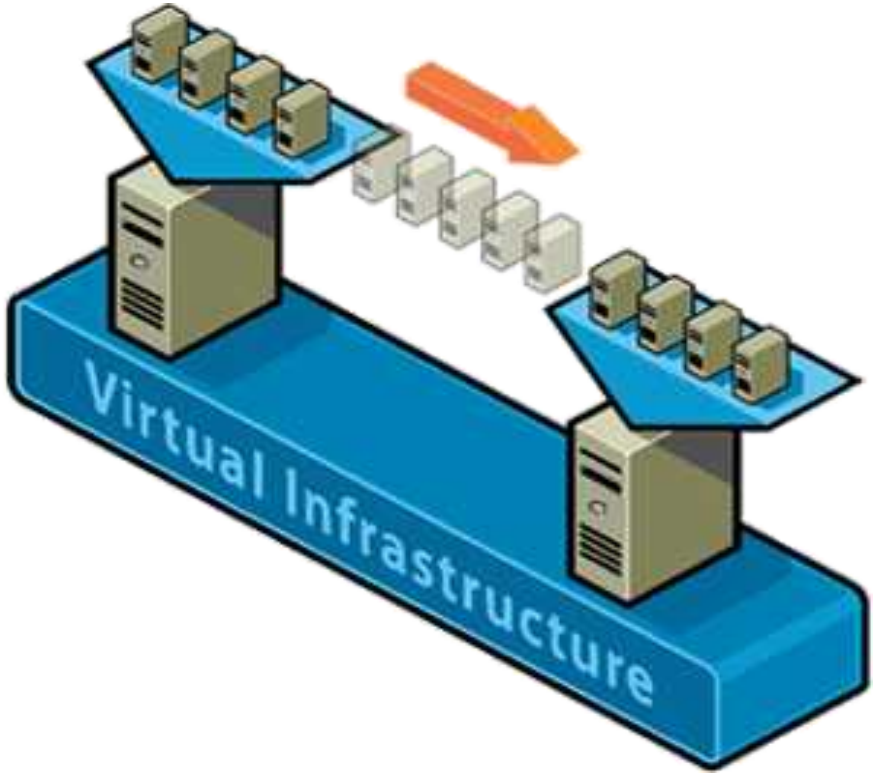


Advantages of virtualization

- Cost reduction
  - 20 to 40% (even if you had a a few tens of servers before and just 2 after)
- More space in the server room
- New functionalities

Virtualization: New features

- Migration of VM from one physical server to the other one
  - In case of failure (rare) → higher availability
  - Maintenance of physical servers (regular, e.g. security patches)



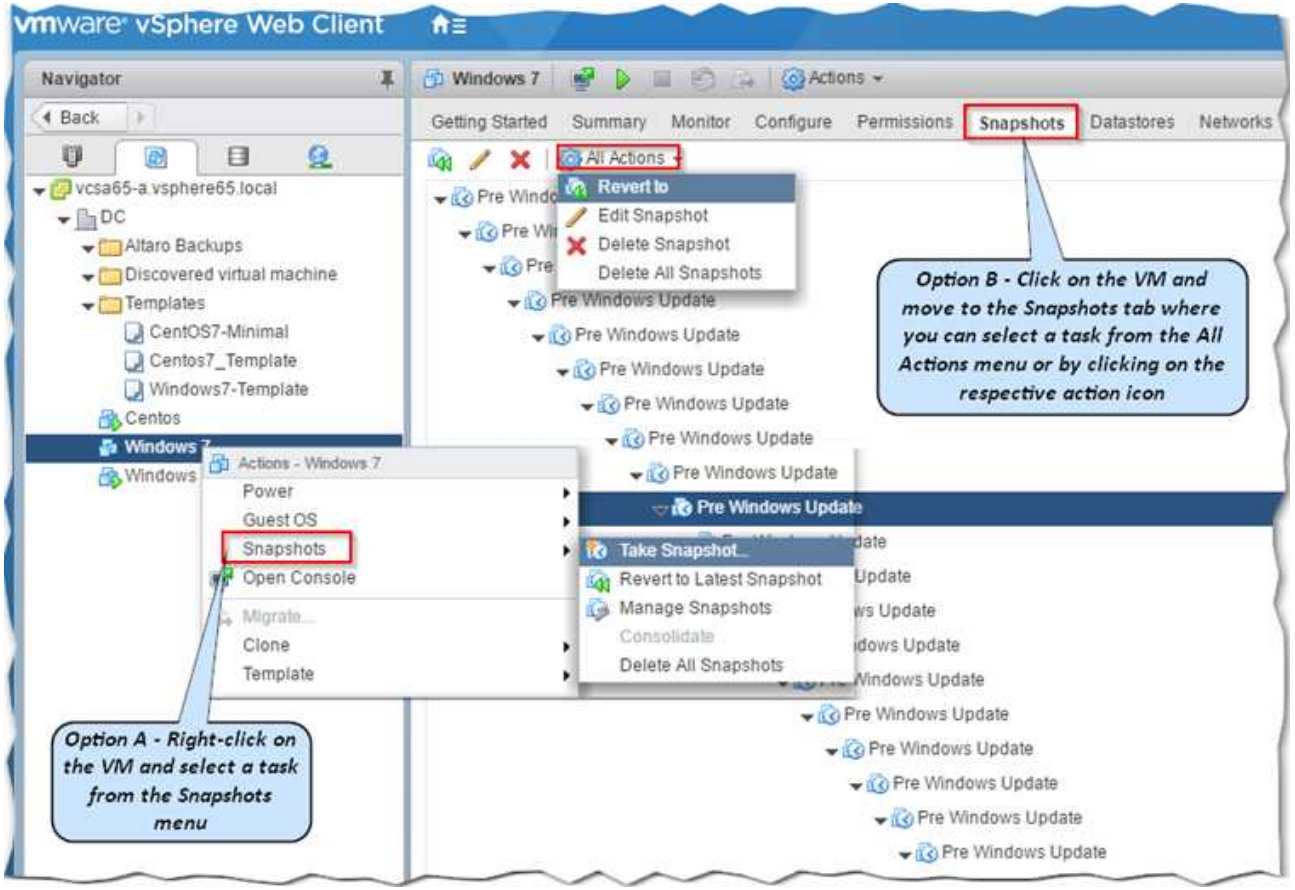
Virtualization: New features

- Fast start-up of new machines through a GUI:
  - Pick number of CPU, amount of memory, disk type and size, network access, OS
  - Indicate where the ISO image of OS
  - Start installation
- Using ISO takes time
  - Use of templates(one for Linux Debian, etc) that are duplicated (aka clone in the virtualization world) on the fly
  - Called images in Virtualization world
    - Vagrant images, AMI (Amazon Images), Docker images
- Current trend : automatic preparation of images out of ISO
  - See Packer https://packer.io/

Virtualization: New features

- Snapshots of VM stage
- Example :
  - You want to update a library/software but you are unaware of final outcome
  - You can :
    - Make a snapshot
    - Install the library/software
    - Roll-back time if it is not satisfactory
  - Also used by hypervisor when freezing (stopping/saving) a virtual machine

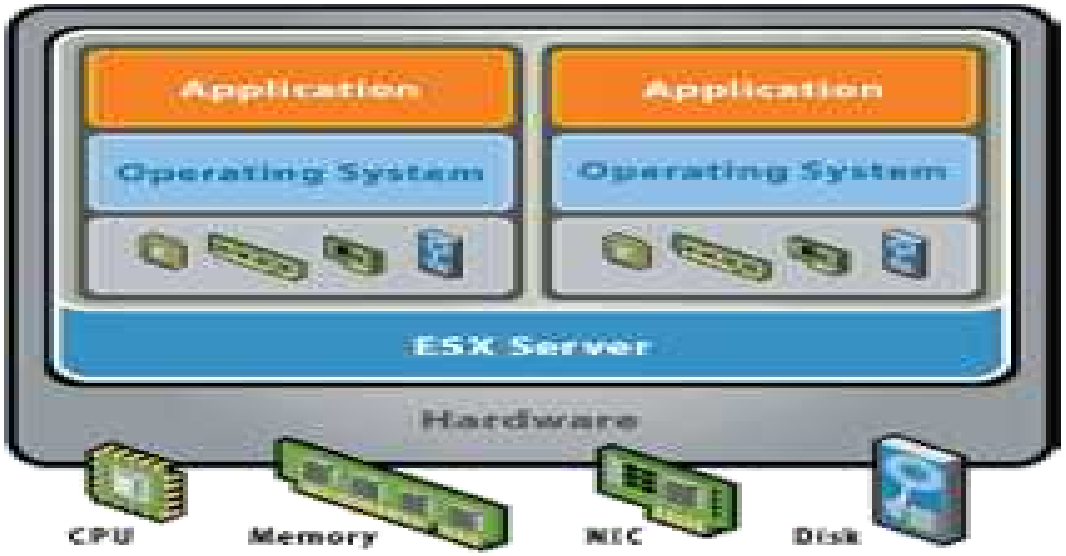
Snapshots



Source : https://www.altaro.com/vmware/wp-content/uploads/2017/02/022817_1130_Workingwith1.png

Virtualization: New features

- Isolation
  - Virtualization still enforces the one server per machine rule
  - If one VM is compromised, the other services remain safe
- On the fly reconfiguration of VMs → more CPU, more memory, new virtual disks
- Easier configuration of VM as hypervisor always displays the same interface to the VM
  - This is the hypervisor that handles the gory details, e.g., supports of SSD drive while VMs are exposed always an SCSI drive → no need to install driver in VMs!!!!



Various types of virtualization

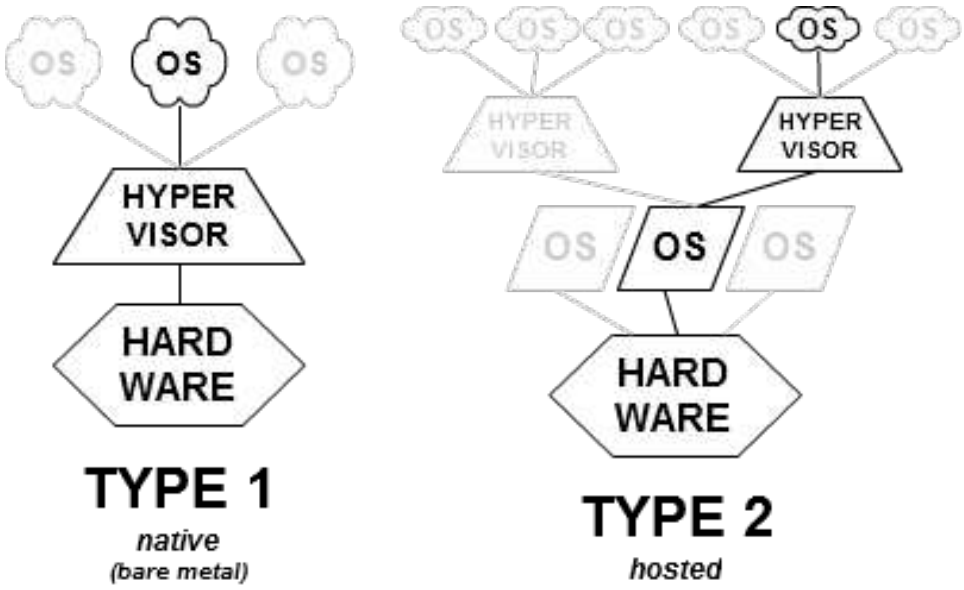
- Different types:
  - Bare-metal (native) versus host-based
  - Virtual versus para-virtual.. versus hardware assisted
  - Container-based versus hypervisor-based

Bare metal vs. host-based

- Bare-metal :
  - Layer 1
  - Production servers, data centers
  - Hypervisor seats directly on top of hardware
    - Machine boots on hypervisor directly
    - Installed as an OS
  - Examples :
    - VMware VSphere Hypervisor
    - Microsoft Hyper-V
    - Citrix XenServer

Bare metal vs. host-based

- Host-based
  - Hypervisor is an application running in an existing OS
  - Layer 2 virtualization
  - Typically deployed on end-user machines
    - VMware player
    - Virtualbox

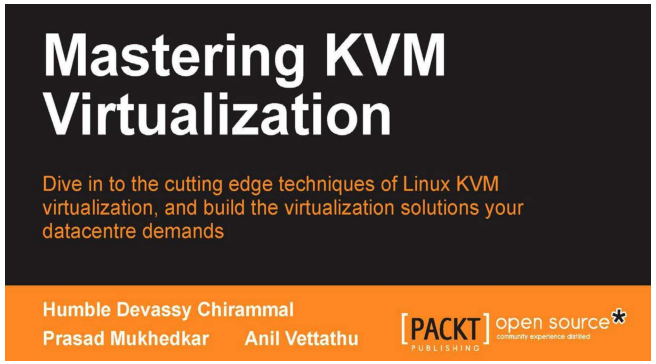


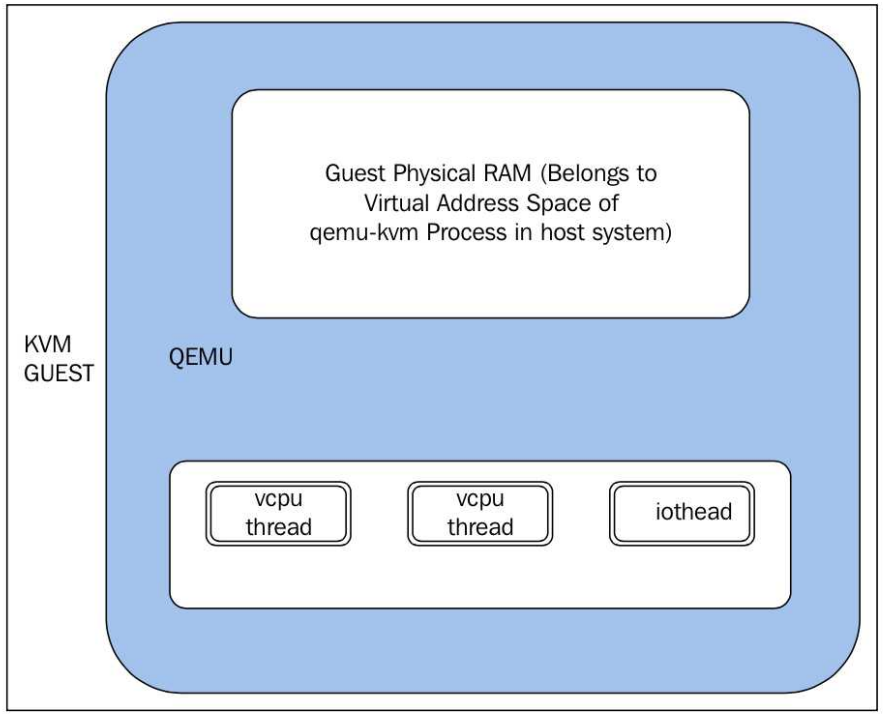
Source : wikipedia

Bare metal vs. host-based

- KVM – Kernel Virtual Machine
  - Support of virtualization in Linux kernel
  - Not an hypervisor per se
- Need QEMU – Quick emulator
  - Layer 1 in terms of perf
  - Layer 2 in style…
- QEMU/KVM Default in Openstack
- Hyper-V similar (hybrid layer 1/layer2)

Source :





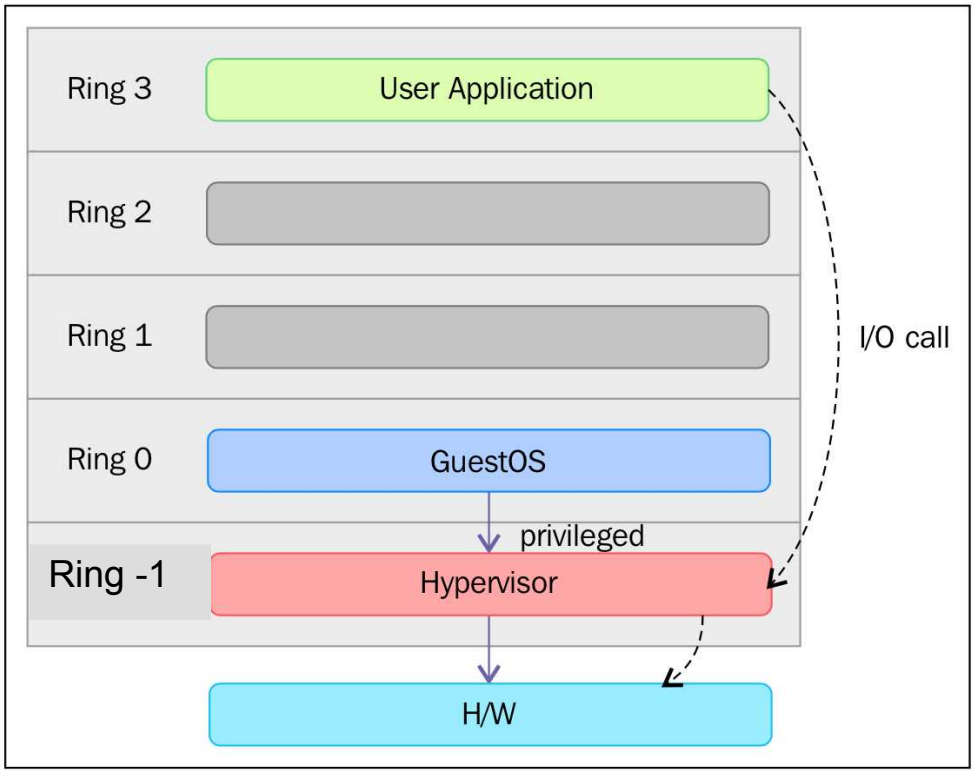
KVM Guest

Full vs. Para-virtulization

- Key question : is there a need to patch the guest OS?
  - No → full virtualization
    - Direct support by hypervisor
    - VMware approach
  - Yes → para-virtualization
    - A (typically small) part of the kernel is patched to interact with hypervisor
    - Better performance
    - Used by Xen initially
- Current trend : no patch but installation of guest additions inside OS

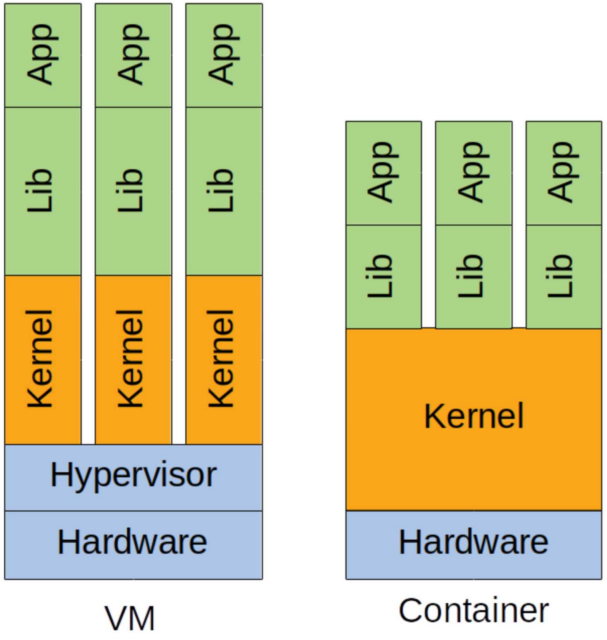
Hardware assisted Virtualization

- Intel and AMD propose various solutions to support virtualization at hardware level → ease of hypervisor task
- Intel VT-x, AMD-V
- Enable OS of virtual machines to do more actions natively → addition of a duplicated structures within processor and a new control level called (-1) for hypervisor



Container-based vs. Hypervisor-based

- Rather than using an hypervisor, the container approach shares kernel among VM
- On a typical server :
  - 10-100 virtual machines
  - 100-1000 containers
- Container engines:
  - LXC (LinuX Containers – August 2008)
  - Docker (started in March 2013)
  - Openvz (started in 2005)



Containers in Linux

- Docker, LXC and the other engines rely on Linux kernel support for containerization
- A container is a group of processes on a Linux host, grouped together in an isolated environment.
  - Use of **namespaces** so as to assign to a set of processes : isolation, their own network stack (interfaces, sockets, routing), volum
  - Use of **cgroups** to assign resources to processes, eg., CPU share, memory limit

Container-based vs. Hypervisor-based

- Containers:
  - Inside the box, it looks like a VM.
  - Outside the box, it looks like normal processes.
  - A container often does not contain a full VM, but application processes, eg. Apache or MySQL server.
- VM:
  - You are really root inside the VM as compared to containers
  - **A "top or ps" command in host OS will display one process per VM with heavy virt. against all the processes inside the containers with light virt.**

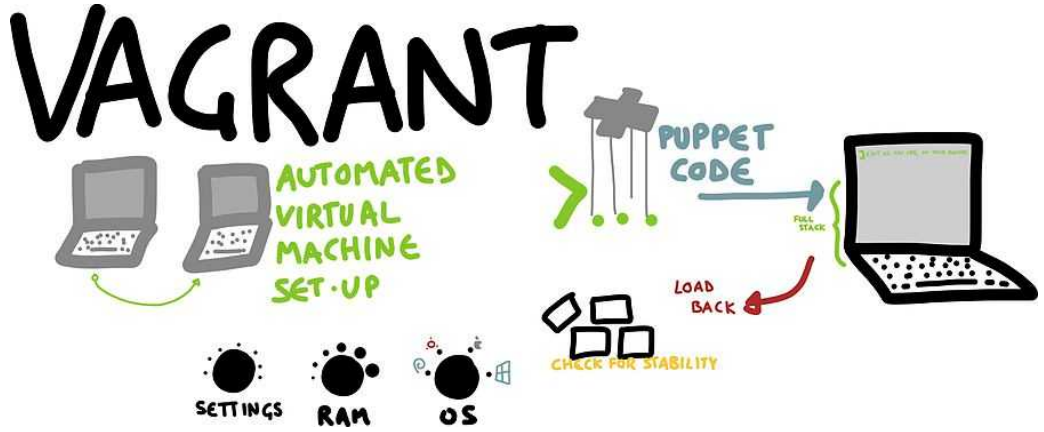
Container-based vs. Hypervisor-based

- Typical arguments for container approach
  - Source : **http://goo.gl/bFHSh**
  - Ships = physical delivery...or download for light/heavy virt

	Ships within ...	**Manual deployment takes ...**	**Automated deployment takes ...**	**Boots in ...**
No virtualization	days	hours	minutes	minutes
Virtualization	minutes	minutes	seconds	less than a minute
Lightweight Virtualization	seconds	minutes	seconds	seconds

Around virtualization: management of VMs, containers, virtual network

- Management of VMs
  - Vmware Vsphere, Citrix Xen are hypervisors and can offer management of a handful nodes of the same type (ESX servers only or Citrix Server only)
  - Vagrant: Management of VMs a hypervisor independent approach
    - Notion of images (boxes in Vagrant)
    - Provisioning of VM: Puppet, Chef, Ansible to configure automatically the VMs
    - A single file that includes everything

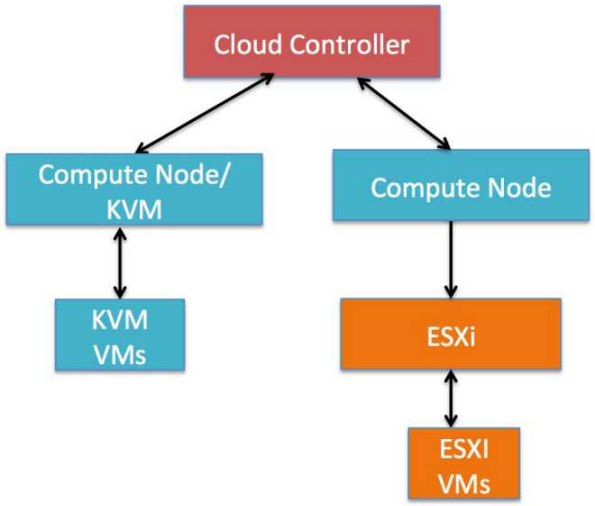


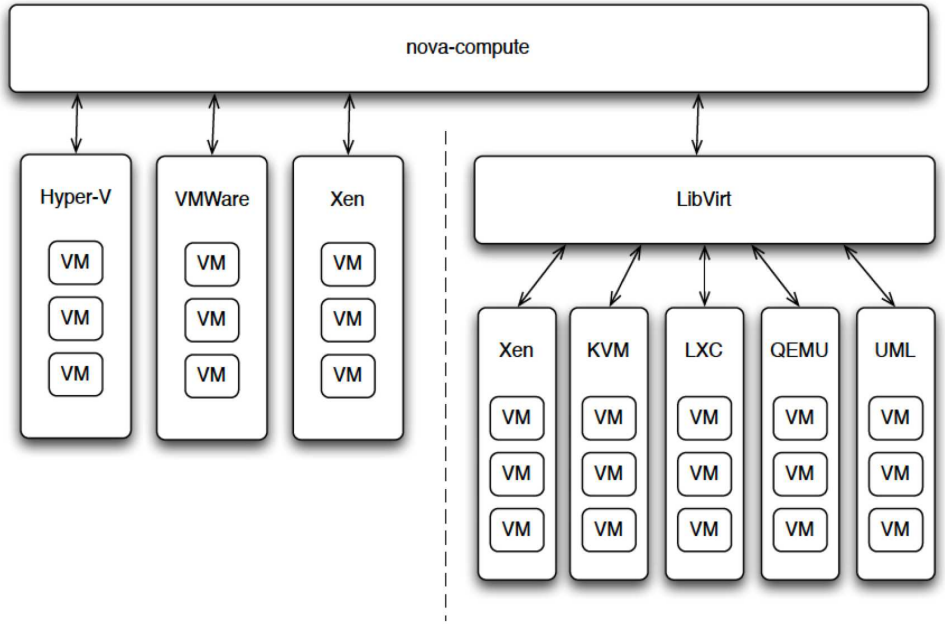
Vagrantfile (excerpt)

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure(2) do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://atlas.hashicorp.com/search.
  config.vm.box = "ubuntu/vivid64"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
   config.vm.network "forwarded_port", guest: 5001, host: 5001

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  # config.vm.network "private_network", ip: "192.168.33.10"

  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
   config.vm.network "public_network"
```

Around virtualization: management of VMs, containers, virtual network

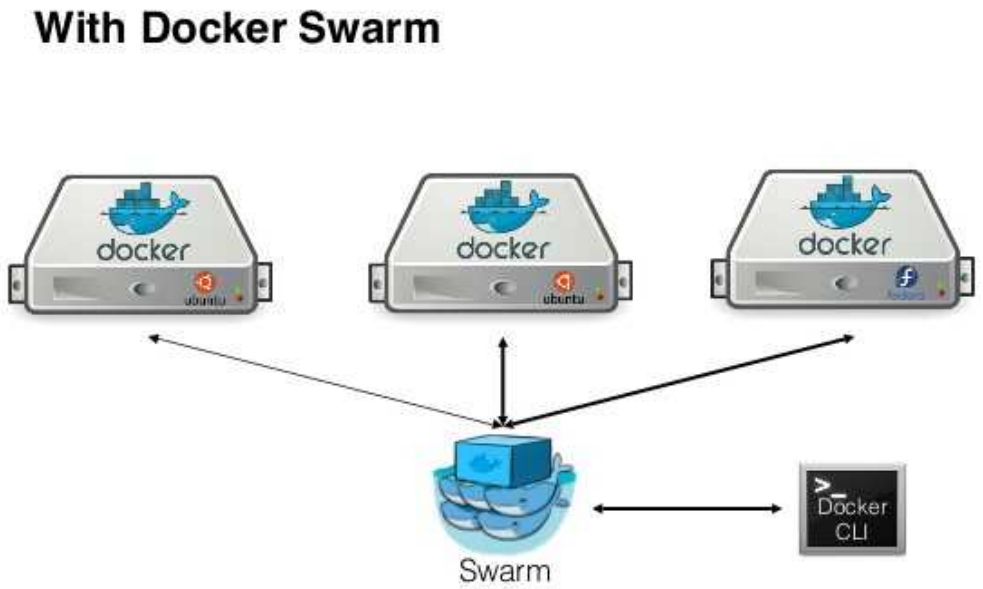
- Cloud platforms to orchestrate at a larger scale, with possibly different hypervisors
  - Openstack
    - Each function (management of VM, of network, of volumes, of identities) is a component
      - Nova: compute nodes (hypervisors)
      - Cinder : volumes
      - Neutron : network
    - Components interact through a REST API
    - Compute nodes (physical servers) might run different hypervisors: KVM, Xen, Citrix, etc





Around virtualization: management of VMs, containers, virtual network

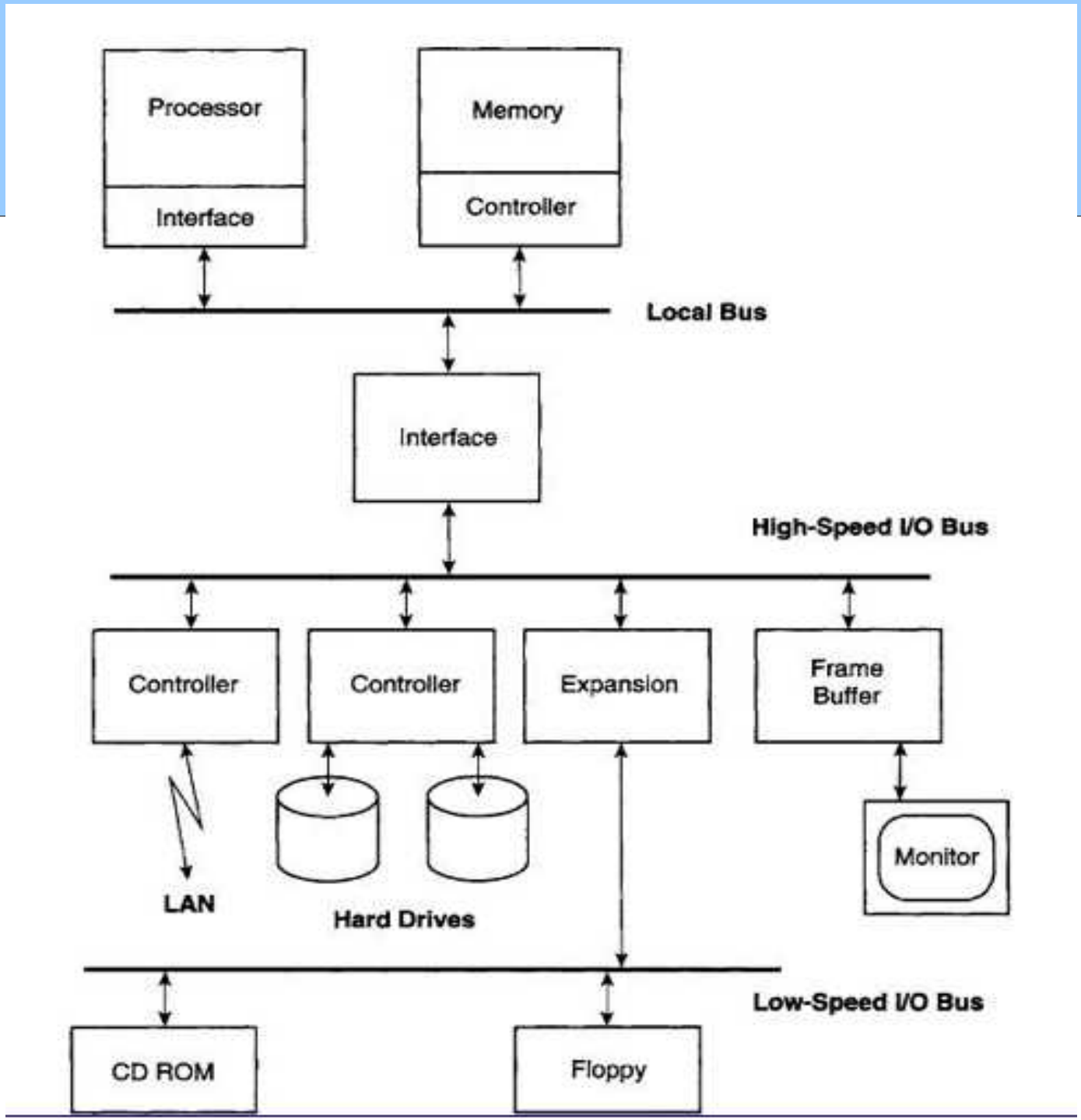
- Orchestration of containers
  - Single server level: Docker, LXC
  - Several servers level: Docker Swarm
- Advanced orchestration: Kubernetes



Part II: the nuts and bolts of (hypervisor-based) virtualization

Refresher on computer architecture and OS

- Computer Components
- ISA (Instruction Set Architecture)
- Operating System


Processor
Interface
Memory
Controller
Local Bus
Interface
High-Speed I/O Bus
Controller
Controller
Expansion
Frame Buffer
Monitor
LAN
Hard Drives
Low-Speed I/O Bus
CD ROM
Floppy

33

Processor

- Nowadays processors
  - Several cores
  - Hyperthreading → 2 processes sharing the same core
- Consequence: if you have p processors with c core and hyperthreading, VMware, Xen and others will expose 2 * p * c vCPU (virtual CPUs)
- We talk hereafter about processor by referring to a basic computation unit (a processor or a core or half a core..)
- Important : at a given time instant, a processor is servicing a single program:
  - Either a user program (processor in user mode)
  - Or the OS (processor in kernel mode)

ISA

- Architecture of a processor defines
  - A set of resources : memory, registers, ...
  - A set of instructions that operate on registers and memory
- Definition of storage resources + instructions → Instruction Set Architectures (ISA)
- One distinguishes :
  - User instructions for programs → computations
  - System instructions for OS → management of resources

Typical ISA: instructions + HW archi

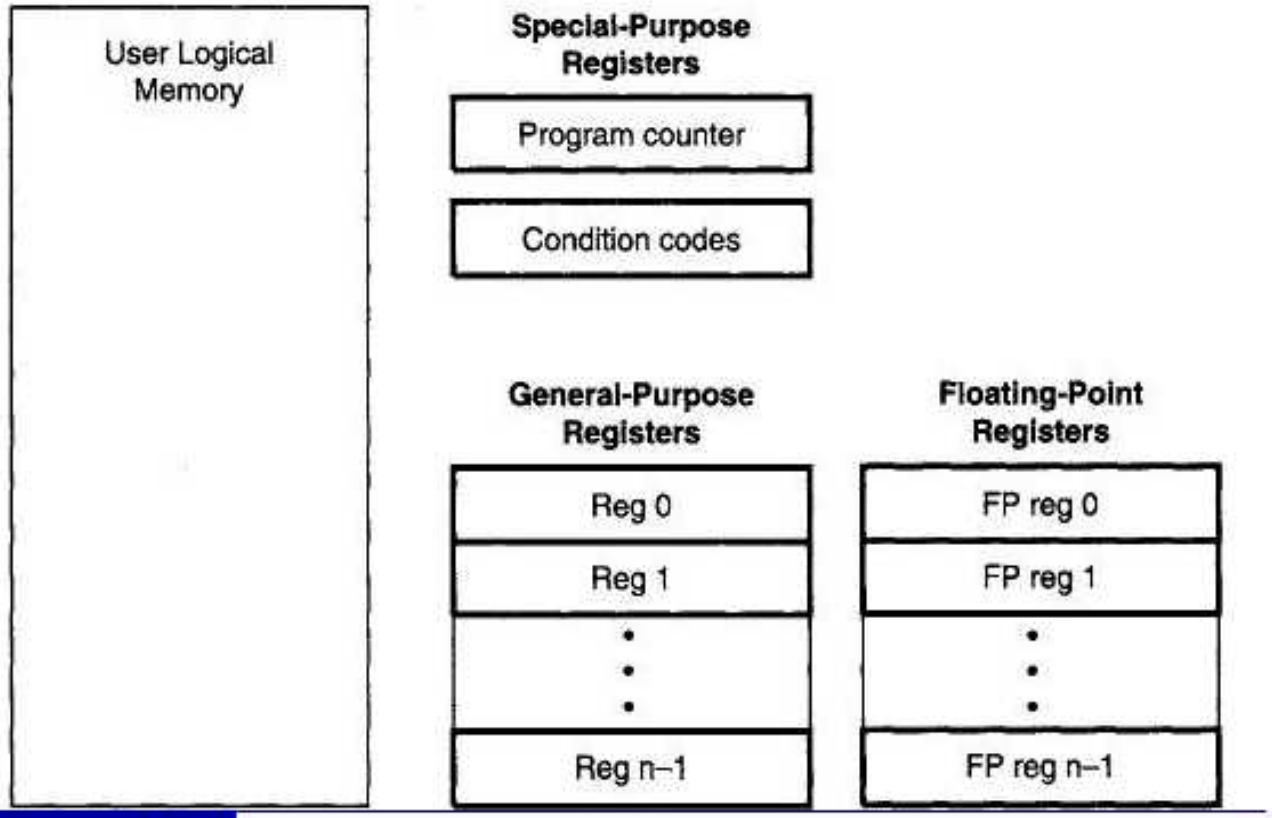


Figure A.6 Key Architected User State of a Typical ISA. *This includes a logical (virtual) memory space, a special-purpose register, general-purpose registers, and floating-point registers.*

Registers

- Generic(aka working) registers
  - Can host different types of values
- Typed registers: for specific operands
  - ISA dependent
  - ex: pointers towards memory segments in Intel IA-32 ISA
- Specific registers:
  - Program counters: index of current instuction
  - Condition

User ISA

- Four main categories
  - Load/store memory ↔ registers
  - Integral/logical operations and shifting
  - Floating point operations
  - Branching and jumps

System ISA

- Processors features several execution modes :
  - In general 4, from 0 to 3
  - Windows and Unix use levels 0 and 3
  - Level 0 : system mod for OS to control and share fairly resources among programs
  - Level 3 : user programs
- Execution mode is stored in a specific register

System registers

- Time Register
- Traps and interruptions registers
- Traps and interruptions masking registers
- Page table pointer
  - Mapping between logical and physical spaces. Kept in RAM
  - “Page table pointer” points to the memory location of this table

Traps and interruptions

- Traps and Interruptions lead to a transfer of control of processor to OS
- Interruption: request from an I/O device to the OS
- Trap:
  - An error during the execution of an instructions(page fault , division by 0, ..)

or

  - An explicit demand from a program

System ISA : management of processor

- OS must be able to hand over control of the processor to a user program
  - Jump to an instruction of a user program
  - Modification of execution state register
- … and must be able to gain control later again
  - Thanks to a timer that will generate an interruption

System ISA : I/O management

- Can be specific instructions
- Can be specific addresses translated as instructions to be executed by memory controller
- Wide variety of I/O devices → ISA offers in general only a few basic instructions
  - OS in charge of communication with devices through the device driver

Refresher on Operating system

OS tasks

- Manage resources on behalf of user programs
  - Time multiplexing of processor
  - Management of physical memory via page table and TLB
    - When page error, OS takes over control of CPU
  - I/O management:
  - Processes perform requests to OS via system calls (that result in traps)
  - OS uses device drivers (that are added to OS) to control devices

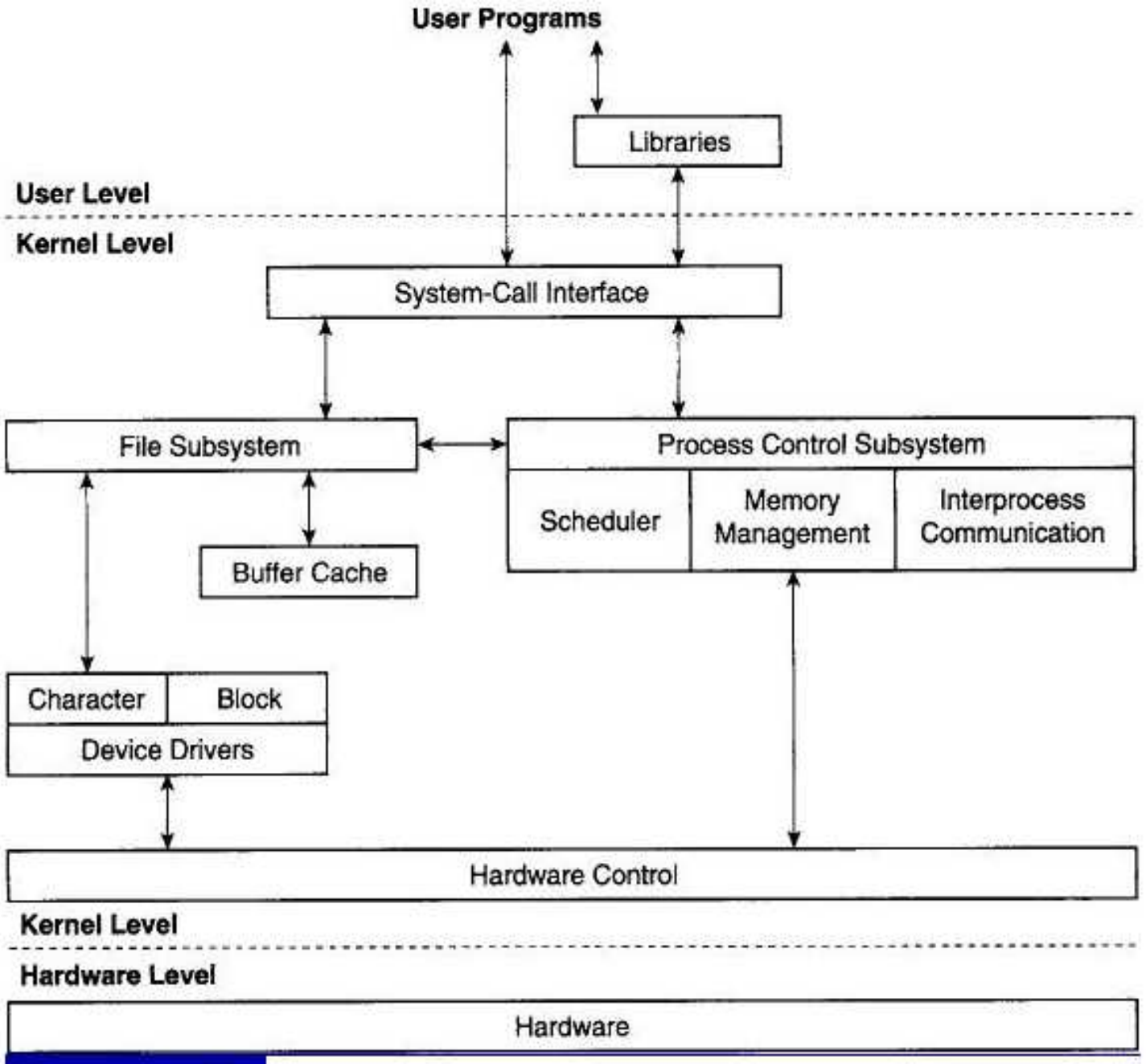


Figure A.12 Linux Architecture.

Interface with OS

- User mode ISA directly accessible to user programs
- ABI (Application Binary Interface) is the interface with OS
- API: high-level libraries (as compared to ABI)
- System calls:
  - Process management, ex : fork()
  - Memory management, ex : malloc()
  - I/O, ex : . read()
- Abstraction of traps and interruptions at ABI = signals

ISA, ABI, API

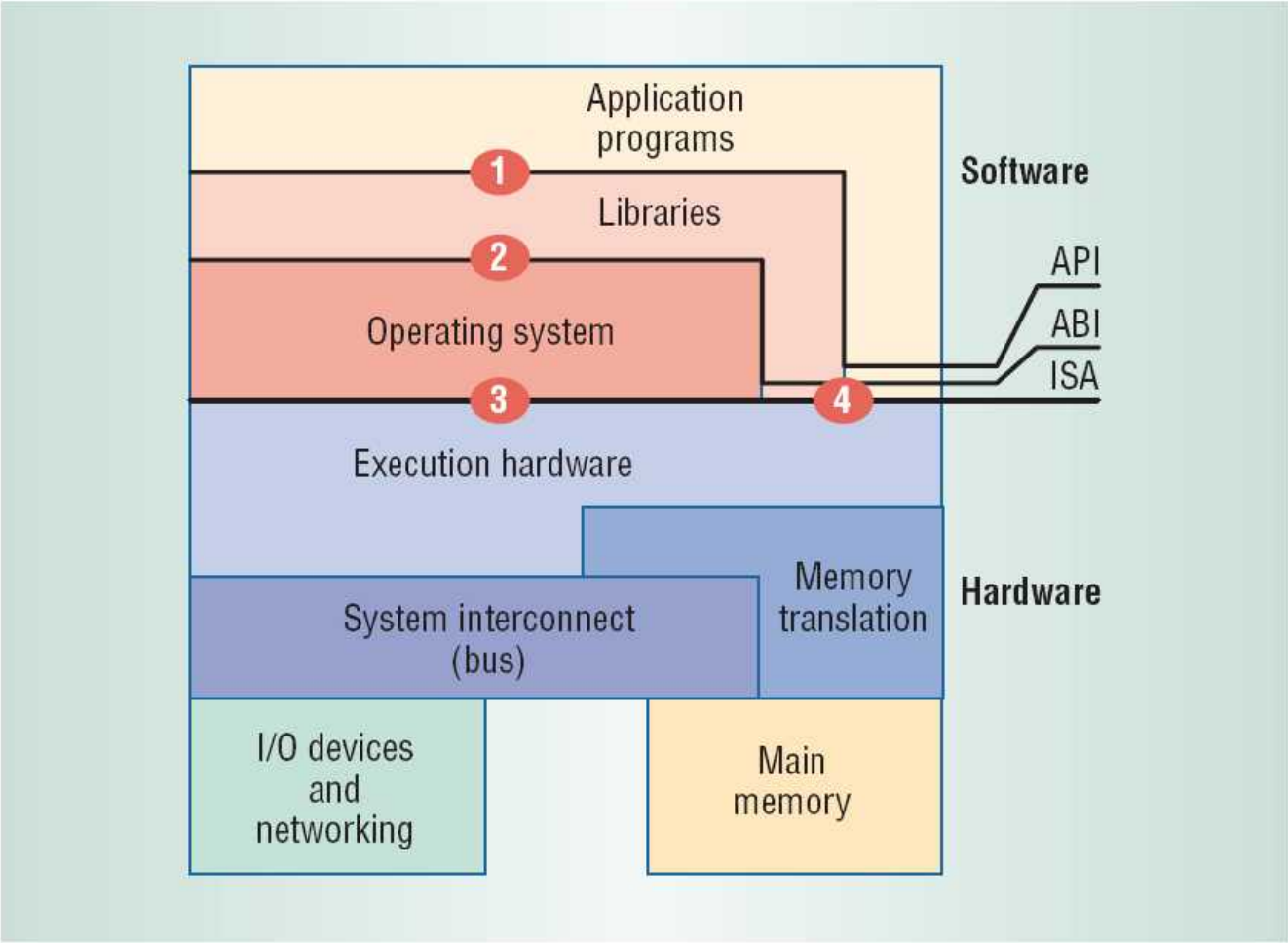


Figure 2. Computer system architecture. Key implementation layers communicate vertically via the instruction set architecture (ISA), application binary interface (ABI), and application programming interface (API).