Redes Software

Grados familia Ing. Telecomunicación

Curso 2024-25

Departamento de Ingeniería Telemática

Pedro A. Aranda Gutiérrez

Universidad Carlos III de Madrid

paranda@it.uc3m.es

Materials taken from the SDN/NFV Master @ UC3M

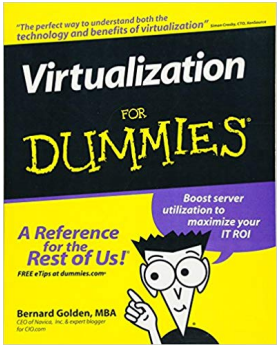
Outline

Software networks: overview of the course

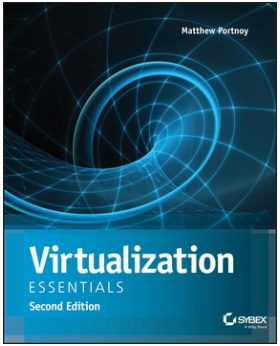
- PART I: Introduction
- PART II: Virtualisation
- PART III: Software Defined Networks
  - Introduction to Software-Defined Networking
  - OpenFlow
  - P4
- PART IV: Network Functions Virtualisation
  - Introduction and motivation
  - NFV architecture
  - VNF software architecture

Useful references
For today...

- “Virtualisation for dummies”, Bernard Golden, John Wiley and Sons Ltd , 2007



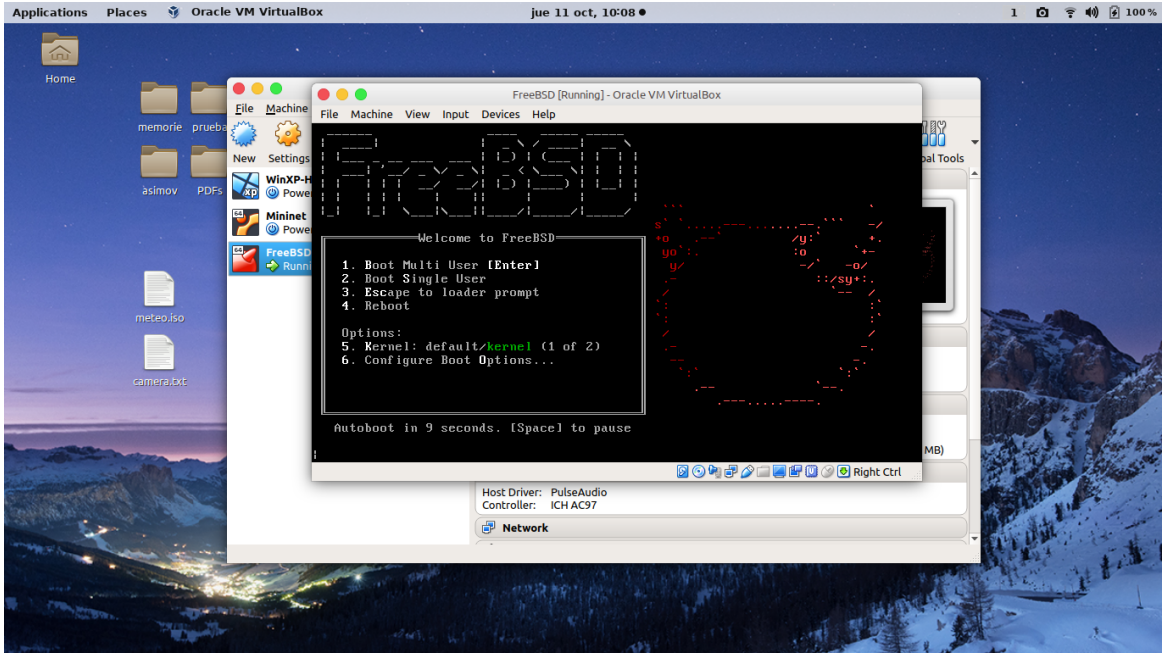
- “Virtualisation essentials”, Matthew Portnoy, John Wiley and Sons Ltd , 2012



- Books available online through UC3M’s library

Outline

Virtualisation



Virtualisation
Definitions

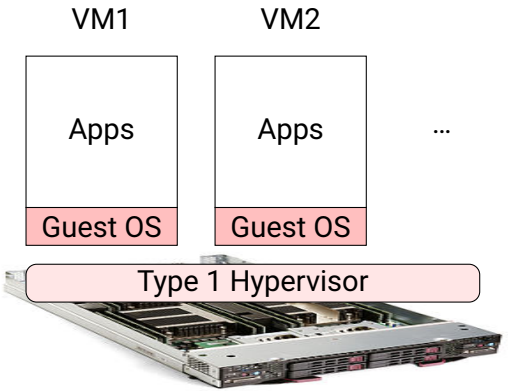
- Virtualisation
  - The abstraction of a physical component into a logical object
  - Access to a single underlying piece of hardware, like a server, is coordinated so that multiple Operating Systems can share it *without* being aware that they are actually sharing anything at all
    - A guest Operating System (OS) is a OS hosted by the underlying virtualization software layer on a host system
- Hypervisor
  - The software providing the environment to abstract the physical component into the logical object
    - aka Virtual Machine Monitor (VMM)
  - Must exhibit 3 properties
    1. Fidelity: The environment created for the Virtual Machine (VM) is essentially identical to the original physical machine the software would execute
    2. Isolation: The VMM (and only the VMM) must have complete control of the system resources
    3. Performance: There should be little or no difference in performance between the VM and its physical equivalent

Virtualisation
Definitions - II

- A hypervisor is a layer of software located (somewhere) between
  1. The hardware, and
  2. The virtual machines that it supports
- We use
  - Hardware = *host*
  - VM = *guest*

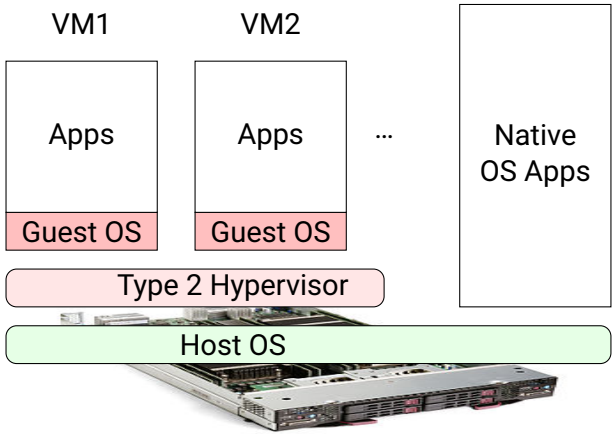
Hypervisor Types
Type 1

- Type 1 Hypervisors run directly on the server hardware
  - Also known as bare-metal implementation



Hypervisor Types
Type 2

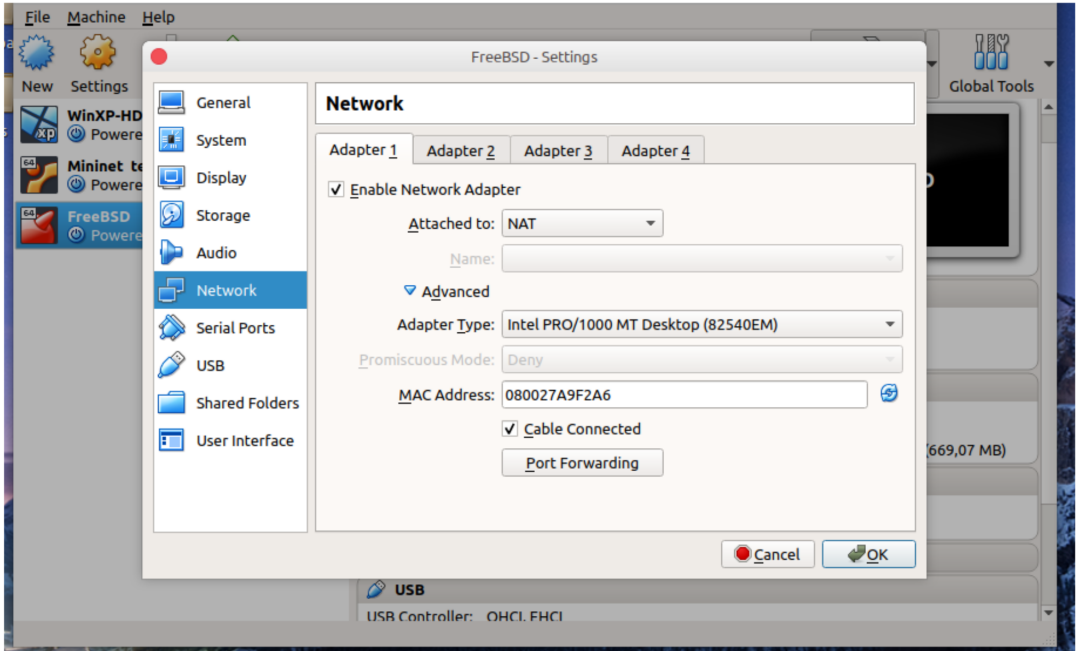
- Type 2 Hypervisors are *applications* that run on an OS



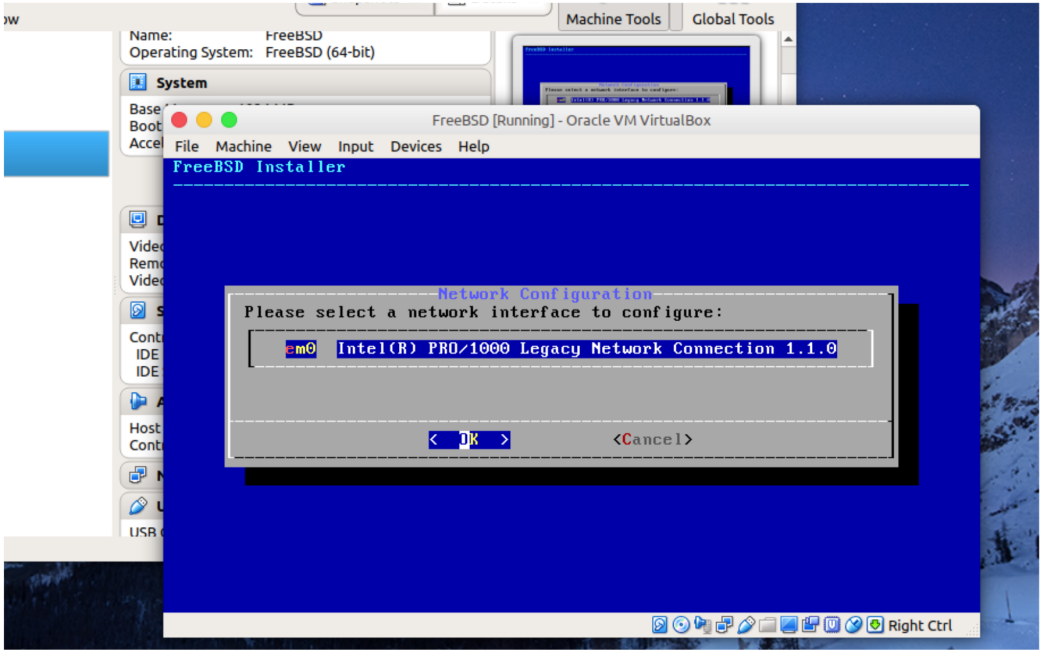
Hypervisors
The hypervisor and the VMs

- The hypervisor presents the virtual machines with generic resources they can use
- Virtual machines have access to what they see as hardware resources, which are actually virtual resources
- Once again:
  - Everything is provided by the hypervisor
  - VMs see “standard” devices, which are actually virtual
  - This makes VMs portable across various hardware platforms

Virtualisation
What Vbox says it provides



Virtualisation
What the VM sees

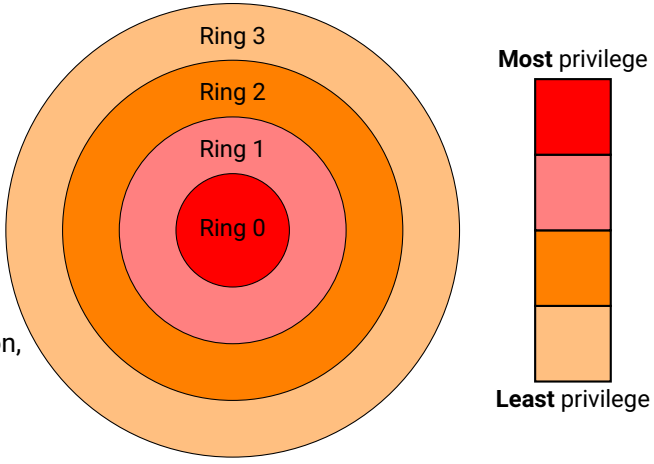


Virtualisation Techniques
Protection rings

- Protection rings are hierarchical protection domains
  - Improve fault tolerance
  - Protect from malicious behaviour
    - thus providing computer security

- Modern processors can theoretically support 2^{64} bytes of memory
  - That is $2^{34} = 17.2 * 10^9$ GBytes
  - So when a program hits an unpopulated memory position, what should we do?

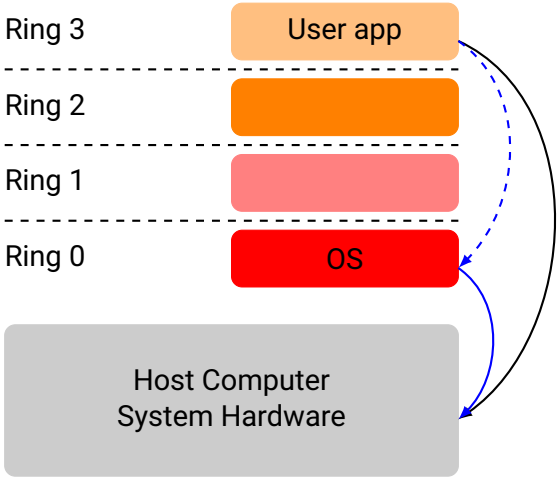
1. The user program is running in Ring 3 and the VMM and OS run in the inner rings
2. When the program requests an "unpopulated" address, there is a *trap* and the OS or the VMM take over.



Virtualisation Techniques

How do we virtualise hosts?

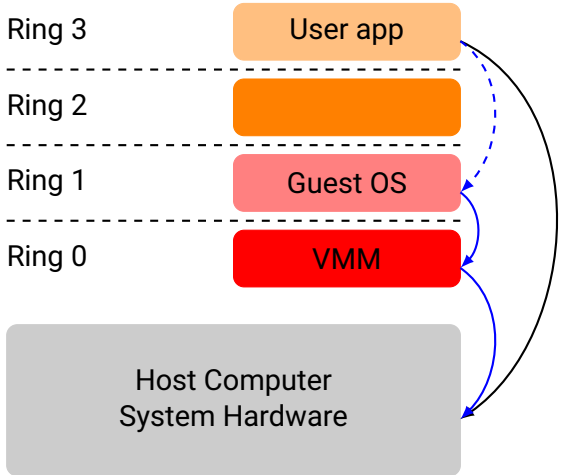
- Modern Operating Systems use the rings to isolate apps from the hardware
  - Applications run in Ring 3
  - The Operating System functions run in Ring 0
    - Access hardware, modify memory management parameters, ...
  - OS system calls *raise* privileges from Ring 3 to Ring 0
- The protection rings are the basis for virtualisation
  - The idea is to place the hypervisor in the most protected ring
  - and the guest OS and software in different rings
  - providing a similar protection scheme



Virtualisation Techniques

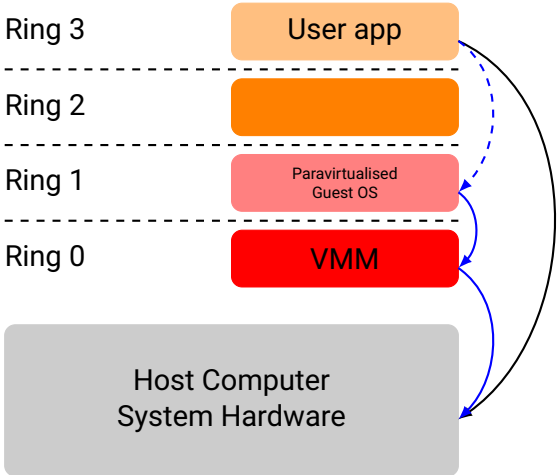
Full virtualisation

- Kernel runs in Ring 1 (not Ring 0)
- User level code is directly executed on the processor
- Kernel code is translated: instructions that cannot be virtualised are translated
- Translated instructions have the intended effect on the virtual Hardware (HW)
- Good portability and portability because the guest OS is unmodified
- Difficult to code all mechanisms needed



Virtualisation Techniques
Para-virtualisation

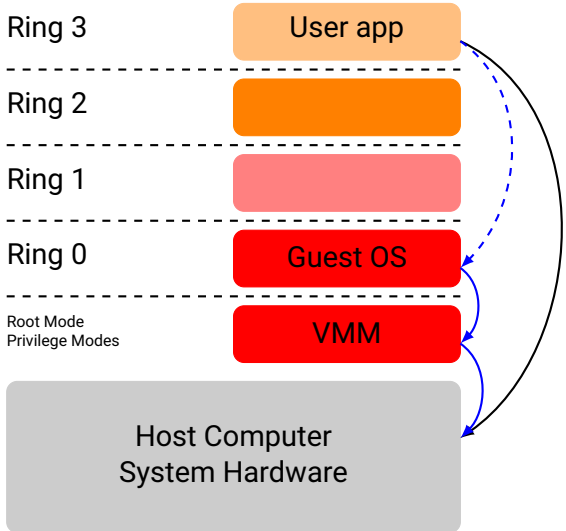
- aka OS assisted virtualisation
- The guest OS kernel is modified by the OS supplier
- Instructions that cannot be virtualised are replaced with hypercalls
- Compatibility and portability are poor, because not all OSes can be modified, however
- Modifying the Guest OS to enable para-virtualization is relatively easy



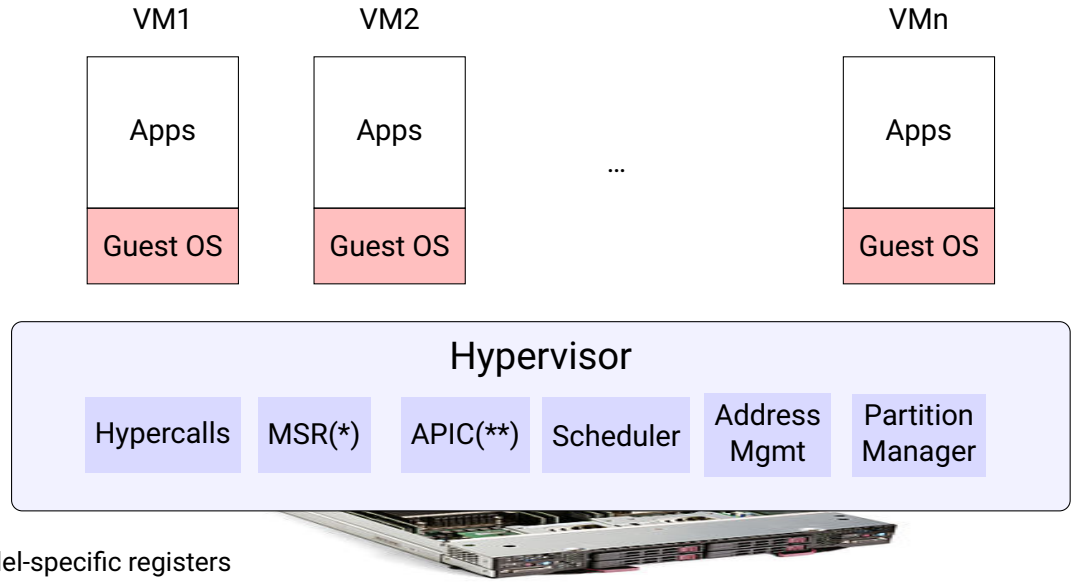
Virtualisation Techniques

Hardware assisted virtualisation

- Modern processors include extensions that simplify virtualisation techniques
- e.g. Intel virtualization Technology (VT-x), AMD (AMD-V)
- Introduces a new root mode that is more privileged than ring 0
- Early generations are quite rigid and binary translation out-performs hardware assist
- Main use case: 64-bit guest Operating Systems



Virtualisation
What is inside the Hypervisor?



(*) Model-specific registers

(**) Advanced Programmable Interrupt Controller

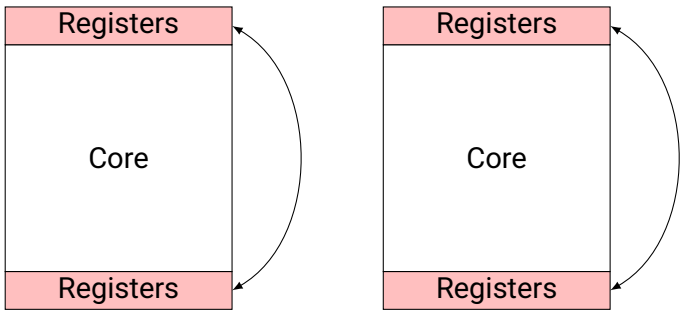
Outline

Virtual Machines
Virtual Machines

- Modern Central Processing Units (CPUs) have multiple cores
- Each core is presented as a single Virtual CPU (vCPU) to the VMs
- A VM may have been assigned more than one vCPU
  - The hypervisor has to schedule physical CPUs among VMs as vCPUs

Virtual Machines
Cores vs. hyperthreads

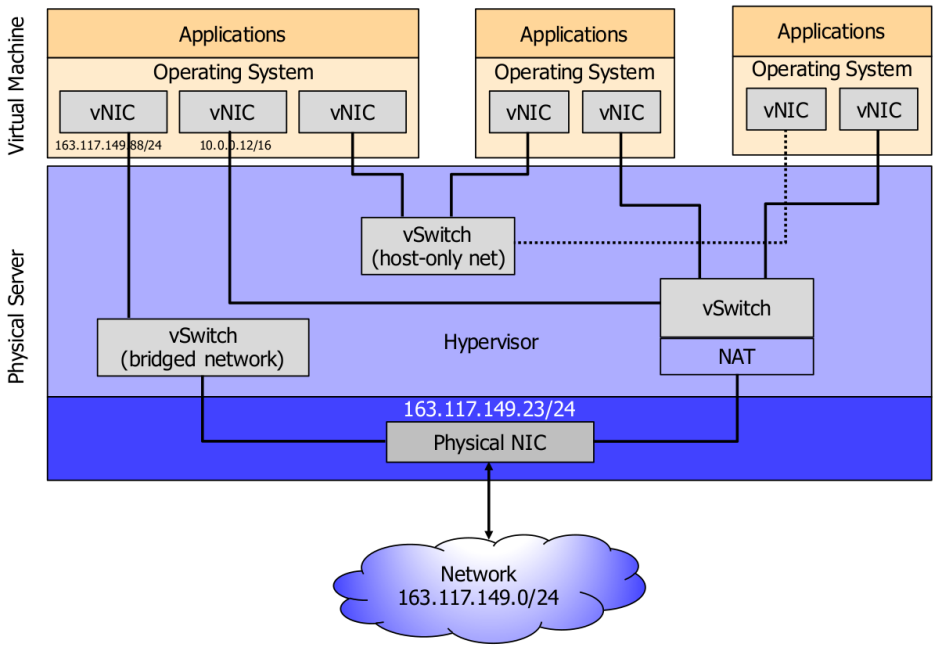
- A core is a full CPU
  - that uses up space in the Silicon die
- In some processors, there is space for two cores, but not for 4 (e.g. i3, i5 processors from Intel)
- Manufacturers create cores with two full sets of registers
  - And make the core switch between them
  - The result is that it looks as if you had a CPU with 4 cores
  - This is called Hyperthread



Virtual Machines
Networking

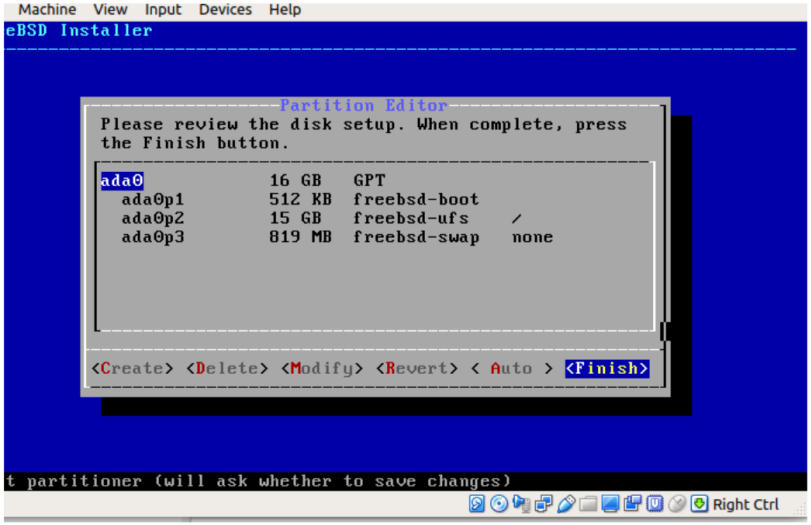
- Each VM can be configured with one or more Virtual Network Interface Card (NIC)s)
- The hypervisor supports the creation of a virtual networks that connect the vNICs to one or more networks composed of virtual switches
- Physical NICs connect to these networks
- The hypervisor normally provides Network Address Translation to some or all of these networks

Virtual Machines
Networking



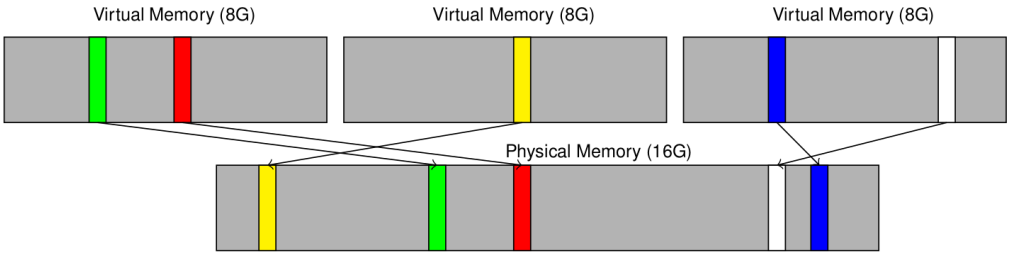
Virtual Machines
Storage

- Storage is also virtualised
- The hypervisor presents virtual storage to the VM
  - Normally a mapping to physical file
  - Exceptionally a physical device directly (e.g. a CDROM)



Virtual Machines
Memory in virtualised environments

- Hot-add memory
  - Adding memory dynamically
    - Initially intended for adding physical memory to servers
    - Quick win in virtualised environments
  - Modern OSes support hot-add memory
  - However, removing memory is not supported
- Memory over–commitment
  - Allocate more virtual memory on a host than physically exists.
  - Why can this be done?
  - Because under normal conditions, not all memory is used:



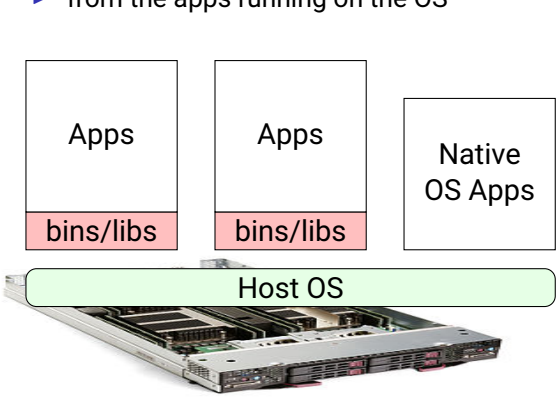
Outline

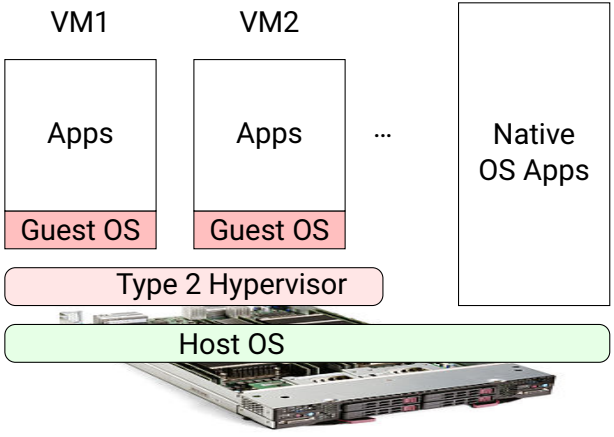
Containers
Introduction

- The kernels of most operating systems provide a way to isolate the users in a machine
  - What gets isolated are the different user-space instances
- Why not just use different users in a machine?
  - Because they share too many things
- Containers: use the facilities provided by the kernel to run different virtual environments
- Is a container like a Virtual Machine (VM)
  - Almost
    - Each container gets its own network interfaces (and networking)
    - …its own filesystem
    - …isolation in terms of security
    - …isolation in terms of resource usage (quotas)
  - At the same time …
    - They are much lighter than real VMs
    - …because they share the same host operating system

Containers
Hypervisors??

- Wait a sec…haven't we seen this already?
- Isn't this like a type 2 hypervisor ?
- Actually not, we neither have a hypervisor …nor a Guest OS as such.
- We just have a thin layer to isolate
  - the apps running in the different containers
  - from the apps running on the OS





Containers
Examples

- LinuX Containers (LXC)
  - Namespaces
  - Cgroups
- OpenVZ
- FreeBSD jails
- Solaris Containers

Containers
Namespaces

- Provide an isolated view of the global resources to processes within a namespace
- Used to support the implementation of containers
- Currently, Linux implements six types of namespaces:
  - Mount namespaces (disk)
  - UTS namespaces
  - IPC namespaces (communication between processes, pipes)
  - PID namespaces (process isolation)
  - Network namespaces
  - User namespaces

Containers

LXC

- LXC allows the execution of Linux applications within a Linux host
- Inside the container, it looks like an isolated Linux
- Outside the container, it looks like a normal process

Containers

LXC: a demo

```
$ lxc-create -t download -n alpine -- -a amd64 -d alpine -r 3.12 \
  --flush-cache --no-validate
$ lxc-start -n my-container -d
$ lxc-info -n my-container
$ lxc-ls -f
$ lxc-attach -n my-container
my-container$ apt -y install apache2
my-container$ ip address show
my-container$ exit
$ lynx $my-container-ip-address
$ lxc-stop -n my-container
```

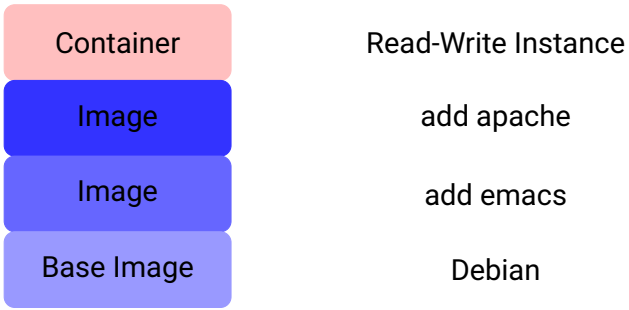
Containers
Docker

- Docker is a wrapper for LXC
- Terminology
  - Docker image: all the files that make up a software application
    - Each change that is made to the original image is stored in a separate layer
    - Each time you commit to a Docker image you are creating a new layer on the Docker image
    - the original image and previous layers are not affected
  - A Docker container is the read-write layer on top of the image

Containers

Docker: images, layers, ...

1. You start with a base image
2. And add features (layers) to create new images
3. And when you execute an image, it becomes an instance



Containers
Docker registry

- A Docker Registry is a public repository where users can download from and upload images
  - Image sharing
- Operations:
  - search for the Docker images
    - `sudo docker search busybox`
  - download a Docker image
    - `sudo docker pull -a busybox`
  - run a shell in a Docker container
    - `sudo docker run - i -t ubuntu:14.04 /bin/bash`
  - run a Docker container in the background
    - `sudo docker run -d apache2`
  - check all containers
    - `sudo docker ps`
  - pause a container
    - `sudo docker pause 466ab23f11a5`
    - `sudo docker pause berserk wescoff`

Containers
Docker: dockerfile

- Define containers using a Markup Language

```
##########################################
# Dockerfile to build an apache2 image
##########################################
# Base image is Ubuntu
FROM ubuntu:14.04
# Author : Dr. Peter
MAINTAINER Dr. Peter <peterindia@gmail.com>
# Install apache2 package
RUN apt -y update && \
apt install -y apache2
# Set the log directory PATH
ENV APACHE_LOG_DIR /var/log/apache2
# Launch apache2 server in the foreground
ENTRYPOINT ["/usr/sbin/apache2ctl" , "-D" , "FOREGROUND"]
```

Containers

Docker: dockerfile

- Build the Container using docker build
  - `sudo docker build –t apache2 - < apache2`
- This allows to boot one container
  - `sudo docker run - i -t apache2`
- However, complex scenarios involve several separate containers
  - Web server frontend
  - Web server backend using a Database
- We need to orchestrate these containers

Containers
Docker: docker-compose

- Using the YAML markup language. Wordpress example [1]

```
version: "3.9"

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
  wordpress_data: {}
```

Containers

Docker: Using docker-compose

- Start dockerised Wordpress on local computer
  - `sudo docker-compose up -d`
- Browse the web-page
  - `lynx localhost:8080`
- Stop the deployment
  - `sudo docker-compose stop`

Containers
LXD

- LXD is a “next generation system container manager”
- Open Source project founded and currently led by Canonical Ltd
- Built around a REST Application Programming Interface (API)
- Relationship with LXC
  - Not a rewrite of LXC
  - Builds on top of LXC:
    - Uses LXC through liblxc to create and manage the containers
- An alternative to the tools and distribution template system used in LXC
- Controllable over the network

Containers

LXD features

- Secure by design (unprivileged containers, resource restrictions, . . .)
- Scalable (from containers on a laptop to thousand of compute nodes)
- Intuitive (simple API and Command Line Interface (CLI))
- Image based (many Linux distributions published daily)
- Support for Cross-host container and image transfer
- Advanced resource control (CPU, memory, network I/O, block I/O, disk usage and kernel resources)
- Device passthrough (USB, GPU, Unix character and block devices, NICs, disks and paths)
- Network management
- Storage management
- Integration with OpenStack

Outline

Just some examples…

- Intel Virtualisation Technology for Directed I/O (VT-d)
  - Hardware assisted remapping
    - Device isolation
    - Improve reliability and security
  - Direct assignment of devices
    - Improve I/O performance and availability
- Single-Root I/O Virtualisation (SR-IOV)
  - You need to understand the fastest bus in your computer
    - PCI Express (PCIe) [2]
  - The hypervisor creates virtual adapters for VMs
  - SR-IOV moves the overhead to the I/O adaptor
  - Physical Functions (PFs) and Virtual Functions (VFs)
    - One Physical Function with up to 256 Virtual Functions
    - The hypervisor configures VFs via the PF
    - PF and VF are PCIe functions

Next sessions

- Hands-on
  - Virtualisation lab
    - Have you installed your HyperVisor software?
      1. VirtualBox
      2. UTM for Apple Mac
    - Remember that you will be building your own lab environment
- SDN
- SDN Labs
- Network function virtualization

Outline

Bibliography I

[1] *Docker docs: Sample applications - Wordpress*. 2013. URL: https://docs.docker.com/samples/wordpress/.

[2] *What is PCI Express (PCIe) - Everything you need to know*. URL: https://www.pcguide.com/tips/what-is-pci-express/.