

Cloud computing deployment and management

First colloquium . 29.11.2021

Efrain Fernandez Sangrador (Erasmus+ Student)

1. Introduction to Cloud Computing

Cloud Computing is an **access** model which:

- on the easy way and upon request
- gives access to sharing of configurable computer resource
- resources can be quickly assigned and free (with minimal interaction with provide server).

Like an **Architecture** promotes accessibility and consist on five basic features:

- self service on request
- wide network access
- pooling of resources (agrupación de recursos)
- fast elasticity
- measuring the services provided

NIST **service** models:

- software in the cloud as a service (SaaS)
- platform for the cloud as a service (PaaS)
- infrastructure for the cloud as a service (IaaS)

Deployment models:

- private cloud
- community cloud
- public cloud
- hybrid cloud

Technologies:

- fast broadband networks
- powerful, inexpensive server computers
- high performance virtualization for hardware

examples: Gmail, Wikipedia, Facebook, twitter...

With hardware virtualization support on commodity hardware, shared elastic distributed computing accounting re.emerged on the consumer (services) market.

Main commands in Linux

sed: Search and replace a text

awk: Filtering files

tail: Print the lastest 10 numbers

wc: word count

diff: compares the differences between two files

nc: netcat, to access to our TCP/UDP ports

ps: Process status

ls: List

mkfifo: Create a pipe FIFO

cat: Concat and print in standard output

tee: Copy the standard output to a file

echo: Printing text in screen.

N01 . Linux, OpenSSH (to make cypher communications)

Cloud computing technologies will be covered, including:

- Message Passing Interface (MPI),
- Docker,
- Hadoop, and
- Google App Engine (GAE)

These enable cloud computing stacks. Specifying intro to facilitating cloud computing will be provided:

- virtualization
- networking
- including a definition of the cloud computing term as stemming from the computer security perspective.

Then, an example will be futher elaborated for Hadoop and VirtualBox,

- demonstrating how to deploy and manage a new cloud of virtual computers
- that can be provided as a service to clients
- using the MapReduce programming model
- for parallel execution in the cloud.

2. Parallel and Distributed Computer System Models

2.1. Heterogeneous Computer System

It refers to electronic systems that use a variety of different types of computational units.

Computational Units:

- A general-purpose processor (GPP)
- a special-purpose processor
- a co-processor
- Examples: DSP (Digital signal processor), GPU (graphics processing unit), ASIC, FPGA.

In general, a heterogeneous computing platform consists of processors with different instruction set architectures (ISAs). We'll have CISC (Complex Instruction Set Computer, that normally they finish like a lot of RISC instructions) and RISC (Reduced Instruction Set Computer).

In the past, the advances in technology and frequency scaling allowed the majority of computer applications to increase in performance without requiring structural changes. While these advances continue, their effect is not dramatic as other obstacles such as the memory-wall and power-wall come into play.

Now, with these constraints, the primary method of gaining extra performance out of computing systems is to introduce additional specialized resources.

Independent computing resources, most heterogeneous systems need considered parallel computing or multi-core (computing) systems.

Hybrid computing is a form of heterogeneous computing where asymmetric computational units coexist with a commodity processor.

2.2. Parallelization of CPU Operations

Superscalarity

Some instructions can execute independently:

- arithmetic (EX)
- loading and saving (IF, ID, OF, WB)
- branching (JMP=)

Restrictions of pipelining:

- data dependency,
- control dependency,
- resource conflict,
- output dependency.

2.3. Multiprocessor Organization

- Single Instruction, Single Data Stream (SISD)
- Single Instruction, Multiple Data Stream (SIMD)
- Multiple Instruction, Single Data Stream (MISD)
- Multiple Instruction, Multiple Data Stream (MIMD): A set of general purpose processors, simultaneous execution of various instruction sequences and different sets of data.

Processing methods of communication:

- Symmetric MultiProcessing (SMP)
 - Two or more similar processors
 - Common memory, bus and I/O devices
 - communication through shared memory
 - symmetry: all processors can perform the same set of operations
- NUMA (Non-Uniform Memory Access) systems
 - An alternative to SMPs,
 - all processors have access to the entire memory
 - memory is divided into segments (variable access times)
 - coordination is different from SMPs and cluster architectures
- **cluster** architecture

2.4. Tasks of the Control Unit

- Concurrent execution,
- scheduling,
- synchronization,
- memory management,
- reliability (fiabilidad)
- fault-tolerance

2.5. Parallelisation of programs with a superscalar Architecture

- The application runs currently on multiple **threads** or **processing units**
- Task of the compiler:
 - Determine independent parts to run concurrently.
 - Division among processors.

- Parallel application:
 - implemented for concurrent execution
 - synchronizes between nodes
 - implementation is more advanced
- Parameterized execution: A task is repeated over different data

2.5.1. Parallel and distributed computer system models (terminology)

Distributed computing is themed in computer science, which addresses distributed systems.

Distributed system is a model of computation, which one's components are located on computers, which are linked in a common network and communicate and coordinate their actions by MESSAGE PASSING (over network).

The components have a common **goal**.

Properties of distributed systems:

- Simultaneous component execution
- don't have a shared clock (because it would be delay)
- components fail individually
- Examples: SOA (Service Oriented Architecture), MMOG (Massively Multiplayer Online Games)

A program on a distribution system is called a distributed program.

Some ways to send **messages**:

- HTTP
- RCP
- message queue

The problems to solve with distributed computing have to be divided into many task, each of which is solved by one or more computers.

Each computational entity, has its own local memory, and communicate with **message passing**.

So:

- Parallel computing: All processors access shared memory (exchange info between processors)
- Distributed computing: Each processor has its own memory (info exchange by passing messages).

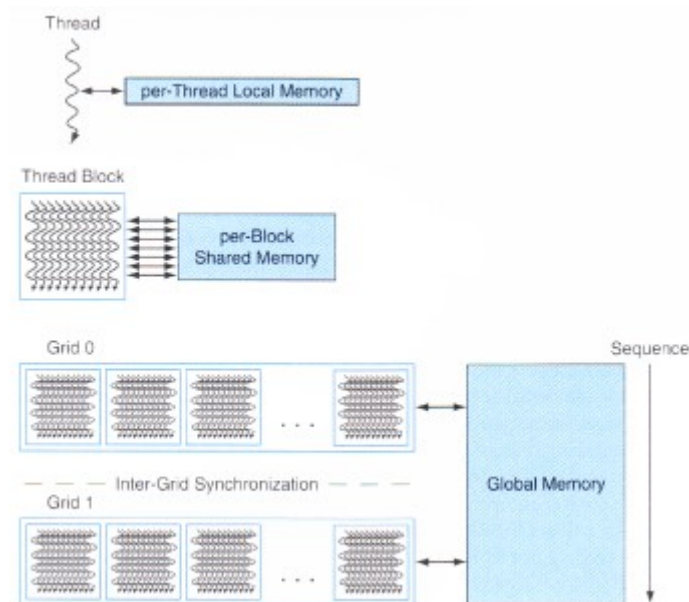
2.6. OpenMP: Multiprocessor Programming

- Parallel programming on CPU using shared memory (most CPU architectures, and Operating Systems)

- Application Programming Interface (API): set of compiler directives, libraries and variables of environment, allows writing cross-platform for multi-threaded.
- C/C++ and fortran.
- Related: MPI, PVM

2.7. CUDA

- Architecture Features
 - GPGPU: General-Purpose Computing on GPU
 - CUDA: Computer Unified Device Architecture
 - A common programming model for general-purpose computing (on the CPU or GPU)
 - NUMA: Nonuniform Memory Access: GPU access memory
 - SIMT: Single Instruction, Multiple Threads: Same instructions execute at multiple threads in a cycle
- CUDA: Thread Indexing.
 - Several hundred threads may execute and communicate (with synchro mechanisms)
 - Kernel threads are grouped in blocks
 - Host computer and CUDA device,
 - program results are same on both: GPU is much faster
 - Independent GPU grids, blocks and threads
 - synchronization
 - Access to memory: cudaMalloc(), cudaFree(), cudaMemcpy()
 - Thread → per-Thread local memory
 - Thread Block → per-Block Shared Memory
 - Grids → Global Memory



A Distributed Computing Example: Web Servers and DNS relay

- DNS resolution request to different IPs.
- Each IP traffic is processed at one computer
- Each computer runs a web server
- An individual task of request handling is a Distributed Task while the larger task of web service to any client is a clustered task.

2.8 Clusters

Is a set of computer resources in numerous locations, related in a common goal. Node allows performing different task or applications.

A cluster can be from a few computers to several thousand and more.

2.9. Grids

A grid computing computers are connected in a network and share sub-tasks. We can create one or more clusters of virtual computers (connected computers that work on the same task together)

Ex: BOINC, [SETI@Home](http://setihome.org)

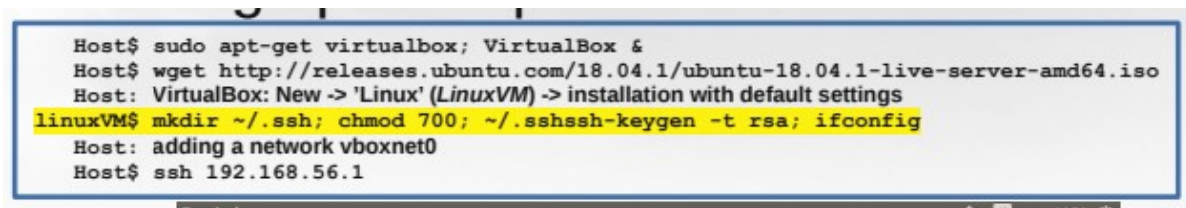
2.10. Peer-to-peer (P2P)

- Any node can be connected to each other
- Nodes act as a client – server at the same time
- Each message can travel through multiple nodes to arrive
- Nodes create virtual connections, and network nodes are announced
- Communication by TCP/IP at the application layer
- Development (desarrollo):

- USENET
- Email: MUA, MSA, **MTA**, MDA
- File sharing: before FTP (File Transfer protocol)+
- Distributed hashing tables (for sharing): in pair the table keeps file ownership for nodes and the nodes keep lists of neighbors (example: **BitTorrent**)

2.11. Clouds

- Instead of sending data for processing, we are sending entire programs (and in doing so we rent hardware, data or software platforms)
- Virtualization of resources (the cloud can host a variety of tasks)
- Rapid renting of clouds
- Redundancy, automatic recovery upon failure
- High scalable, with real-time monitoring
- Setting up a simple virtual enviroment:



```
Host$ sudo apt-get virtualbox; VirtualBox &
Host$ wget http://releases.ubuntu.com/18.04.1/ubuntu-18.04.1-live-server-amd64.iso
Host: VirtualBox: New -> 'Linux' (LinuxVM) -> installation with default settings
linuxVM$ mkdir ~/.ssh; chmod 700; ~/.sshssh-keygen -t rsa; ifconfig
Host: adding a network vboxnet0
Host$ ssh 192.168.56.1
```

We can use Virtual machines for make more PCs in only one.

3. Clustering

3.1. MPP Architectures (Massively parallel processing)

- A task is divided into several subtask that run in parallel.
- Each processor uses its own operating system and main memory
- MPP communicate via **messaging interface**-
- Examples: decision systems, data warehouses

3.2. OpenMPI: Multiprocessor Programming (CPU)

OpenMP enables distributed programming on extension and **OpenMPI** is directed towards distributed programming.

- Acts on sending and receiving messages.
- Node organization is chosen by the program
- Ex: client-server, tree, p2p

3.2. MPI: Message Passing Interface

- Is **distributed**: multiple computers at the same time by pieces works on one task
- MPI work is divided into several processes
- Each node is driven by one (or more) processes
- Each MPI process is run by the same program but with different rank
- The ranks (id) divide the work between nodes

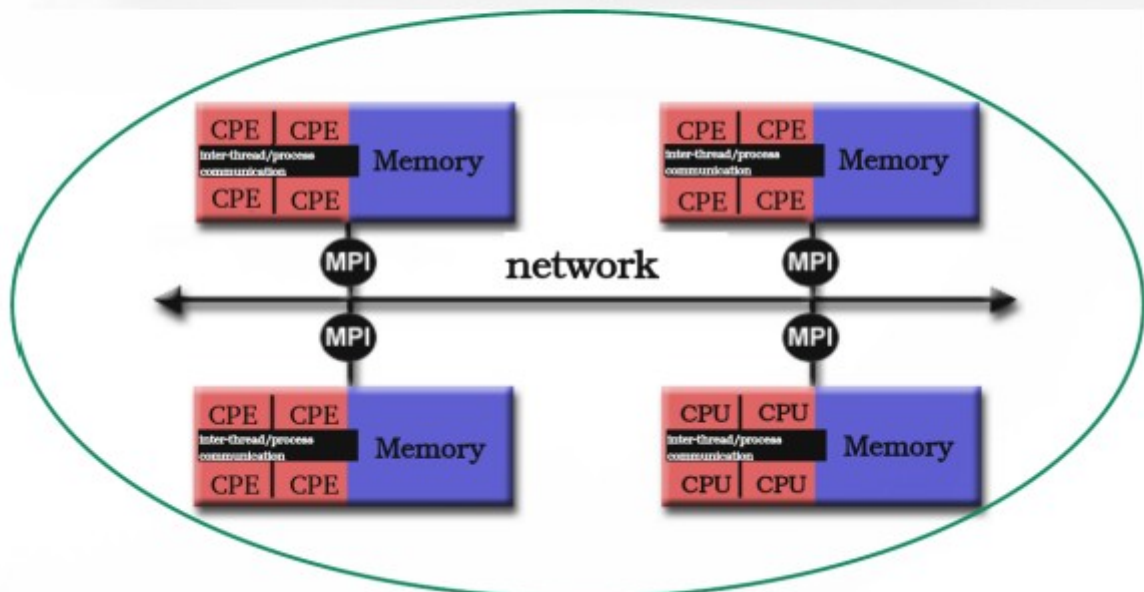
3.3. Comparison: Parallel vs distributed

- **Serially**: One computer processes program piece by piece
- **In parallel**: Multiple processors or computers at the same time by pieces works on one task
- **Distributed**: Multiple computers at the same time by pieces works on one task. More speedup and we solve problems of capacity.
- Comparison **shared** vs **distributed** memory:
 - shared has fastest communication (handled by OS/HW) → OpenMP (Multi-threading)
 - Distributed: Programmer takes care of communication, slower, with delay → **MPI**
- **Pitfalls** of parallelization:
 - Memory access patterns change

- Concurrent (I/O) request
- Additional overhead in the code

MPI → Parallelization with distributed memory

MPI – parallelization with distributed memory



MPI – *Message Passing Interface*

- MPI work distributed into several processes
- Each node is driven by one (or more) processes

4. Virtualization

Enables shared use of expensive machine resources, by defining:

- composed Virtual machines
- virtualization levels,
- VM architectures
- virtual networks
- virtual clusters
- virtual data repositories
- dynamic structures for clusters, networks and clouds.

Three requirements for VMM:

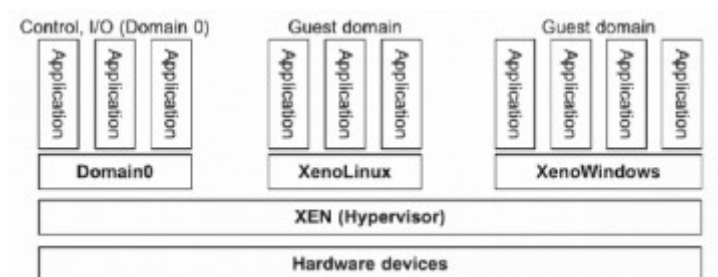
- Provides an environment equal
- Speed is slower
- VMM has full control

Implementation Levels:

- 1. ISA (instruction set architecture), with dynamic ISA translation
- 2. HAL (Hardware Abstraction Layer)
- 3. OS VE: Deployment of a Virtual Execution
- 4 & 5. Process

VM architectures:

- **hypervisor**: Between the hardware level and the OS



- **binary instruction translation**: Complete virtualization
- **para-virtualization**: Changes the OS kernel and **hypercalls**

Cloud Computing and Deployment

2nd colloquium

EFRAIN FERNANDEZ SANGRADOR

1. Docker

First of all, I'm going to explain in Spanish Docker for have a global vision in my language, and later I'll start with the notes in English of the University's slides.

1.1. Inicios en Docker

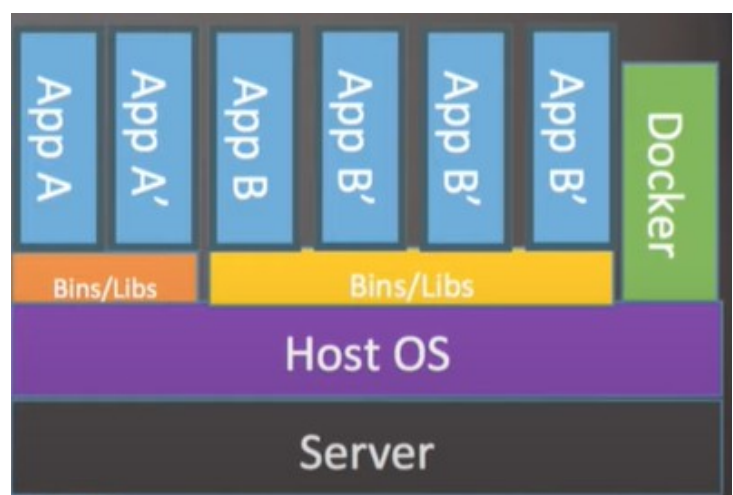
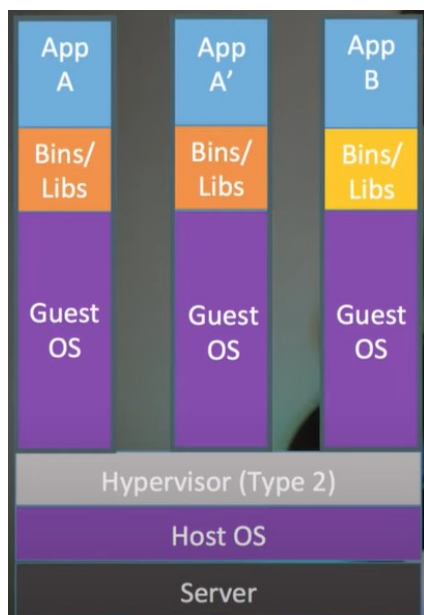
Literature: https://www.youtube.com/watch?v=CV_Uf3Dq-EU (Compleat Course of Docker)

<https://www.youtube.com/watch?v=6idFknRIOp4> (Learn docker in 14 minutes)

<https://www.youtube.com/watch?v=Uu49ID0NBRQ> (What is Docker and how it works)

Es una forma de de containerizar aplicaciones, es decir, de tener encapsuladas aplicaciones completas en contenedores, con sus dependencias y los programas que necesiten. Aplicaciones auto-contenidas.

Comparando con la virtualización: La máquina virtual es un sistema operativo completo. A parte de la app que queremos desplegar, usamos ficheros del SO (gasto de recursos muy superior). Como alternativa, llega el uso de contenedores. Comparten muchos ficheros con el SO base para ahorrar en recursos.



Como ejemplo, descargamos una **imagen** de Mongo DB, y para utilizarlo tendremos que encender un **contenedor** que la ejecute. Para ello: `docker run -p 27017:27017 mongo`

Con ello haremos que se ejecute en el puerto 27017 que es el puerto de mongo DB

Si queremos que se ejecute en estado background haremos: `docker run -d -p 27017:27017 mongo`

Y para matarlo posteriormente haremos: `docker ps` (para ver todos los docker process status)

`docker kill nombrecontenedor` para matar al contenedor.

Contenedor



Un contenedor tendrá en el empaquetado el Sistema Operativo, las dependencias y la aplicación, que estará ayudada por las dependencias y el OS.

Imagen



La **imagen** es el diseño y los planos para instalar el Sistema operativo y las dependencias. Podremos crear una imagen con un Dockerfile o descargar una similar a lo que necesitamos.

En el Dockerfile pondremos las instrucciones de docker para que primero descargue el SO FROM ubuntu

Para crear la imagen: docker build -t ubuntu img [C://ruta_DockerFile](#) (t es el tag)

Y creará la imagen a partir del dockerfile.

Con docker images podremos ver que imágenes tenemos

Para crear un contenedor a partir de una imagen: docker run -it imageId (it será para un modo interactivo).

Crearemos un archivo bastante simple de HTML y meteremos en una imagen de docker un servidor de aplicaciones como apache

Ahora para crear la imagen, haremos el FROM y posteriormente el COPY que sirve para copiar archivos. Con el “.” copiará todos los archivos de la carpeta, a la ruta que le indiquemos dentro de la imagen. Ejemplo:

FROM nginx:1.19.0-alpine

COPY . /user/share/nginx/html

para correr el docker en un puerto, a través de un contenedor: docker run -it -p 80:80 imagenDocker

Comandos

docker images - Ver imágenes

docker ps - Ver Contenedores activos

docker ps -a - Ver todos los contenedores, incluye inactivos

docker system prune - Borra contenedores inactivos.

docker build -t [imagen] - Construye la imagen

docker run -it -p 80:80 [id] - Inicia un contenedor utilizando una imagen

Con esto lo que estará pasando es que habrá un computador interno (el alpine) que estará sirviendo un archivo HTML en su puerto 80 hacia el puerto 80 de nuestra máquina.

Tutorial Docker 2021: De Novato a Pro. https://www.youtube.com/watch?v=CV_Uf3Dq-EU



Ej: OS: linux, software: apache, app: .html

Dockerfiles

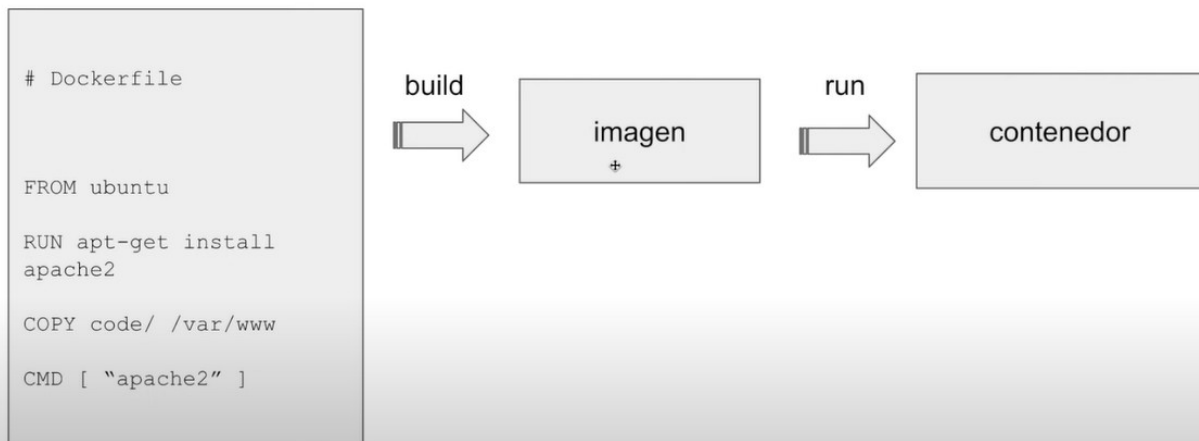


Image from: <https://www.youtube.com/watch?v=QXlQDV9FZhI&t=628s> (How to create the best Dockerfiles)

Podremos usar una imagen que esté en docker hub, y poner un tag para elegir la versión que queramos. Ejemplo docker run postgres:tag

Comandos más comunes de Docker

docker run: Para correr una imagen de docker

docker pull ubuntu: para solo descargar una imagen sin correrla (hacer el contenedor)

docker images: para ver que imagenes tenemos creadas

docker ps: contenedores que estan corriendo actualmente (-a los que están ahora)

Se podrán rescatar contenedores pero normalmente no es recomendable guardar cosas en los contenedores, sino modificar las imágenes.

Para comenzar el contenedor: docker start nombre/id del contenedor (iniciará en background)

docker log (-f) nombre/id_container: Para ver los logs del contenedor

docker exec: Ejecuta un comando en un contenedor que ya está corriendo, mientras que docker run crea un contenedor a partir de una imagen. -it (para que sea interactiva)

EJ: docker exec -it nombre_id_container comando

docker stop id

docker run -d nombre_id

Escribir un Dockerfile (<https://www.youtube.com/watch?v=6ZnecM3ipu4>)

La primera linea siempre será un FROM con el Ubuntu/alpine que vamos a poner.

Si queremos una imagen externa → FROM node:12.22.1-alpine. Normalmente se utilizará la distribución de Linux Alpine, no tendrá apenas espacio ocupado.

Con WORKDIR /app ponemos que todo se ejecute en esa carpeta

Con COPY . . Copiará todos los archivos que están donde el dockerfile en el . Del contenedor, que será el /app

Con RUN yarn install --production corre ese comando

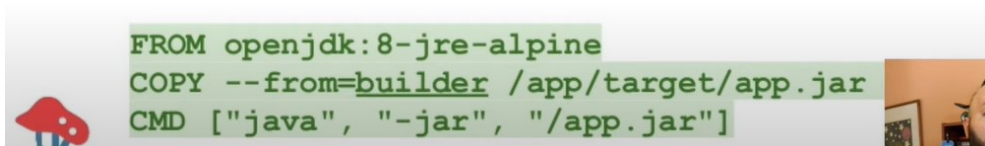
Con CMD["node", "/app/src/index.js"] → Hacemos que se ejecute el comando "Node" con los argumentos en este caso "/app/..." Se puede sobrescribir facilmente

Ej: App de Java: <https://www.youtube.com/watch?v=QXlQDV9FZhI>

Tenemos una carpeta con el proyecto de Java y el Dockerfile. Creamos el Dockerfile con (Siempre mejor el COPY al final porque sino se invalida lo anterior). Siempre será mejor buscar una imagen con buenas prácticas que no instalar un Ubuntu y luego el java. Habrá que instalar además con el tag de la versión que queremos. Se pueden hacer varios stages en Dockerfile

Multi-stage builds to remove build deps

```
FROM maven:3.6-jdk-8-alpine AS builder
WORKDIR /app
COPY pom.xml .
RUN mvn -e -B dependency:resolve
COPY src ./src
RUN mvn -e -B package
CMD ["java", "-jar", "/app/app.jar"]
```



```
FROM openjdk:8-jre-alpine
COPY --from=builder /app/target/app.jar
CMD ["java", "-jar", "/app.jar"]
```

Podremos crear archivos bidireccionales con Docker para guardar bd...

docker run -d -v /Carpeta1:/CarpetaEnDocker -p 3000:3000 contenedor

Con esto podremos cambiar además código, así podremos hacer los cambios y no tener que reconstruir la imagen. Al acabar el día tan solo tendremos que hacer un docker build -t nombre:v2

Con docker push podremos compartir en docker hub nuestros contenedores.

Multicontenedores (https://www.youtube.com/watch?v=CV_Uf3Dq-EU)

Crearemos una red de docker para que se conecten entre ellos

docker network create red

```
kbs@touchbar:~/ejemplo-docker/multi-container$ docker run -d \
> --network todo-app --network-alias mysql \
> -v todo-mysql-data:/var/lib/mysql \
> -e MYSQL_ROOT_PASSWORD=secret \
> -e MYSQL_DATABASE=todos \
> mysql:5.7
```

docker exec -it id mysql -p → Se corre el comando en el contenedor

Para correr un contenedor en esa network → docker run -it --network todo-app aplicación

```
kbs@touchbar:~/ejemplo-docker/multi-container$ docker run -dp 3000:3000 \
> --network todo-app \
> -e MYSQL_HOST=mysql \
> -e MYSQL_USER=root \
> -e MYSQL_PASSWORD=secret \
> -e MYSQL_DB=todos \
> getting-started:v2
```

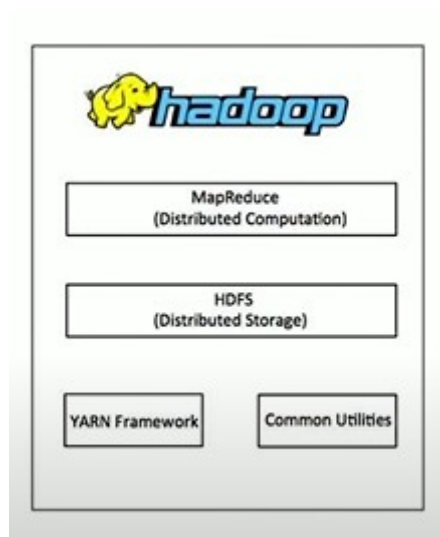
Docker Compose es un programa para mezclar todo en un mismo archivo. Abriremos un archivo “docker-compose.yml”

```
version: "3.7"
services:
  #docker run -dp 3000:3000 --network todo-app -e MYSQL_HOST=mysql -e MYSQL_USER=root -e MYSQL_PASSWORD=secret -e MYSQL_DB=todos getting-started:v2
  app:
    image: pablokbs/getting-started:v1
    ports:
      - 3000:3000
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos
  mysql:
    image: mysql:5.7
    volumes:
      - ./mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos
```

QUE ES HADOOP: Es un framework que permite el procesamiento distribuido de grandes cantidades de datos usando modelos de programación simple sobre un cluster.

<https://www.youtube.com/watch?v=yJsITGEmWm8> – What is hadoop (OpenWebinars)

- Procesamiento distribuido
- Eficiente
- Económico
- Fácilmente escalable: Solo tendremos q añadir nodos.
- Tolerante a fallos
- Open source



- Common utilities: librerías necesarias
- YARN: Gestor de recursos (de las máquinas)
- HDFS: Sistema de archivos, instalado en todos los nodos.
- MapReduce: Paralelización de los datos.

Siempre será un cluster con una máquina maestra y N esclavas.

2. Exam questions

2.1. Cloud definition

Specify NIST definition of a cloud: Cloud computing is an access model, which,

- on the easy way and upon request,
- give access to sharing of configurable computer resource

resources are can be quickly assigned and freed (with minimum overhead at management and minimal interaction with service provider).

What are the characteristics of cloud computing according to the NIST definition: Promotes accessibility and consist in:

- self-service on request
- wide network access
- pooling of resources
- fast elasticity
- measuring the services provided

Give some examples of distributed computing: Netflix, Twitter, Facebook, Whatsapp, Github, amazon, airbnb...

For bash (Linux), explain the operation of piping and streams:

- The pipes are a redirection way to send the output of a program, to another.
- The streams are the different output or input systems of text.

2.2. Questions Seminar N02

How do we unpack material from DK.UM for N02 that contains multiple files? In a folder

Explain the file structure in Linux and assign access rights:

- Is an a tree structure. The higher is the root "/" and the others are below.
- For the access rights, we have the a three number, that is an octal way to say which access rights are activated. The first number indicates the users rights, the second group, and the third, execution. Each of this groups can have read, write and execution permits.

Describe at least five Bash shell commands for working with files and streams:

- cat command: Allows to print in the screen some files
- rm, cp, mv, touch... ls
-

Describe at least five Bash shell commands for working with networks:

- ping: for send packages between devices
- dig: for DNS information
- netstat: Display connection information
- ifconfig: For see the interfaces
- route: for manipulate IP routing table
- hostname: to identify our IP

Which command lists network paths in Linux for routing network traffic and how are these related to service outages FB / IG / WA 4.10.2021?

netstat command?

Explain some process management tools in Linux: you can manage process in linux with ps for see process in execution, kill for kill them and stop for stope.

How do we set up network connections in Linux?: We have to:

- Have a static IP configured
- Check the network information
- Make a ping to another device

2.3. Docker and Singularity containers and scaling

Specity a command in Docker to create a simple container with an Apache server in a Ubuntu distribution: docker image Apache:ubuntu → docker run apache:ubuntu

Specify a command in Docker to scale the container to 1500 nodes: docker scale <container> 1550.

Describe the file structure for the Docker image description: The first of all always is going to be a FROM, later a COPY, RUN or CMD

Explain the orchestration in Docker and the command swarm: The orchestration are the tools to manage scale of containers. With swarm, you initialize Docker swarm mode, you choose one manager node and its all

Which algorithm does Docker use for consensus? Which hypervisor does it use on Windows and how on Linux?

When the docker Engine runs in swarm mode, manager nodes implement the Raft Consensus Algorithm to manage the global cluster state <http://thesecretlivesofdata.com/raft/>

In windows docker uses Microsoft Hyper-V and on Linux the same.

Describe converting an image from Docker to Singularity and then deploying with 300 nodes

docker run -it image

To deploy, with docker scale, swarm or make an script.

2.4. Process-Level Parallelization

List at least five examples of types of computational (process) units:

- A-general purpose processor (GPP)
- A-special purpose processor
- A co-processor
- Digital Signal processor
- GPU

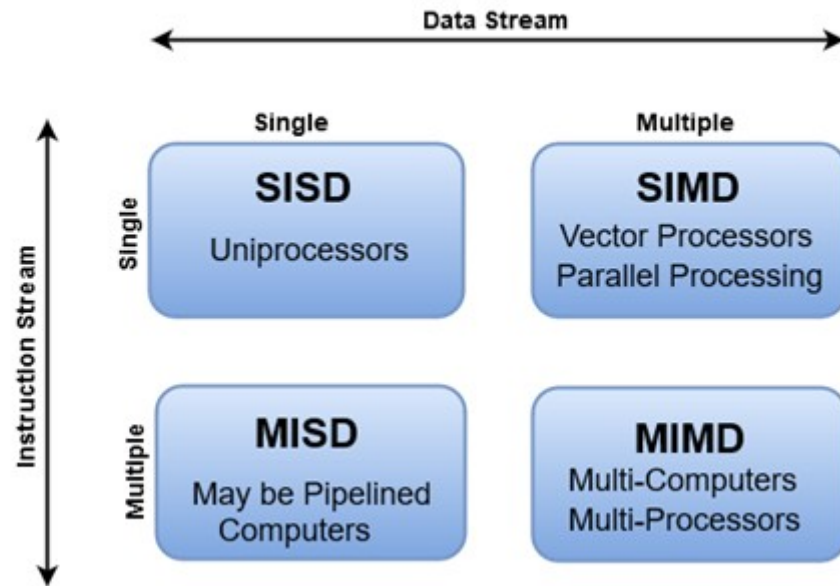
What is Instruction Set Architecture (ISA): Is the CPU part visible to the programmer. Server the different instructions. (RISC, CISC)

When can heterogeneous computing resources be used for parallel computing?: When they have specialized resources to avoid restrictions

Explain ISA command-level pipelining and Flynn organization: Each Instruction pass between different cycles. The pipelining allows to have more than one instruction → parallel.

Flynn organization:

Flynn's Classification of Computers



The last one MIMD a set of general purpose processors. Has for communication: Symmetric MultiProcessing (SMP), NUMA systems and Cluster architecture.

What is NUMA?: Is a Non-Uniform Memory access in where all processor have access to the entire memory, divided into segments.

Explain the tasks of the control unit in the processor:

- Concurrent execution
- Scheduling
- Synchronization
- Memory management
- reliability
- fault-tolerance

Specify some types of program parallelization through superscalarity:

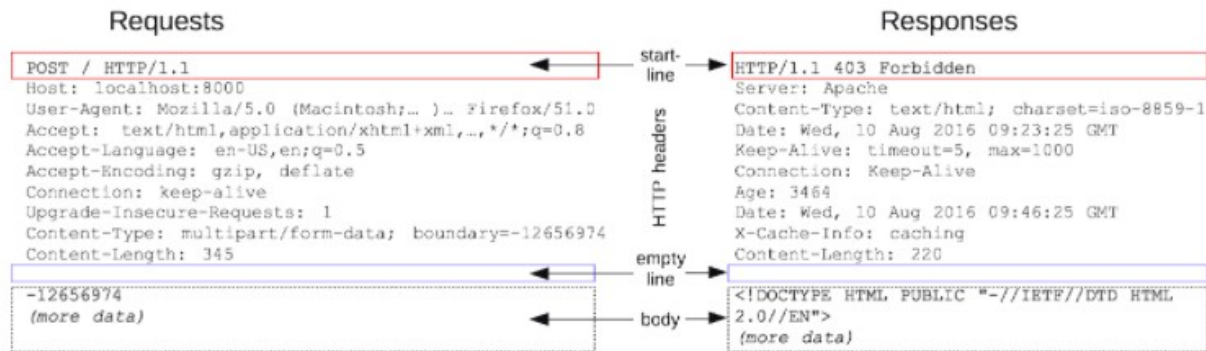
- Application runs on multiple threads or processing units
- The compiler determine independent parts to run currently
- Parallel aplication is implemented for concurrent execution, with synchro nodes

2.5. Operating system level parallelization

What does message sending mean in a distributed architecture?: It means that the different devices in the distributed architecture are linked by messages (over network)

Why do distributed systems not have a common clock? Because with a common clock, they would have delays and would work worse.

Give an example of an HTTP message:



How are local memory and messaging related? But what is shared memory and what is the difference? What is allocated memory?

Local memory and messaging are related by message passing. One device sends request to another, and this another replies with the response.

Shared memory is for Parallel computing, and all the processors can access to the same memory.

Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space.

Does OpenMP allow messaging or multi-threading? OpenMP enables distributed programming on extension. Allow multi-threading.

Why do we need a queue if we use messaging? Because without it we could lose messages and not respect the orders

2.6. CUDA, server distribution

Explain the terms CUDA and SIMT:

- **CUDA** (Computer Unified Device Architecture): A common programming model for general-purpose computing on the CPU (host/serial) and GPU (device, co-processor/parallel).
- **SIMT** (Single Instruction, Multiple Threads): Several same instructions execute at multiple threads in a cycle.

What is kernel in CUDA?: A kernel is a function that executes in N different threads instead of sequentially. The execution of a kernel in multiple threads (threads) is organized "spatially and temporally" as a grid (mesh) of thread blocks.

What is a block in CUDA?: Is a number of threads working in the same task

What do we call the shared memory between threads in CUDA for the same block?

Describe the syntax for accessing data from a block loop:

```
int i = blockIdx.x * blockDim.x + threadIdx.x
```

Specify the program syntax for the declaration, which part of the CUDA code will be executed on the GPE device?

`__global__` or `__device__`

Specify calls in the program code to transfer data to and from the CUDA device

`cudaMemcpy` → three types (DeviceToHost, HostToDevice, DeviceToDevice)

What is a distributed task? What is the difference between distributing CUDA and DNS routing to entire computers?

A distributed task is an individual task of request handling, involving more than one computer and it is split in subtasks.

CUDA is one task with multi-threading and the DNS is one task with more subtasks.

2.7. Types of Partition-to-Cloud Distributions

Explain the terms process, task, node, and cluster:

- process: Is an instance of a program
- task: A unit of execution
- node: One device/Operative System
- cluster: a set of nodes working together

Explain the concept of network computing and give some examples of communication with each other: A network computing is a network with more than one computer connected, and where they exchange information.

Examples: Peer-to-Peer: Any node can be connected to each other (node act as client and server at same time).

- E-mail (MTA)
- File sharing: FTP
- Distributed hashing tables (P2P)

On the other hand, clouds send entire programs.

Give some functionality of network management systems and explain how they are connected to the cloud: Network managements systems are providing functions to control, plan, locate, deploy and monitor network resources.

They are connected to the cloud by a service provider.

2.8. Examples of architectures through evolution

Give some examples of simultaneous operation of programs on Linux (users, processes):

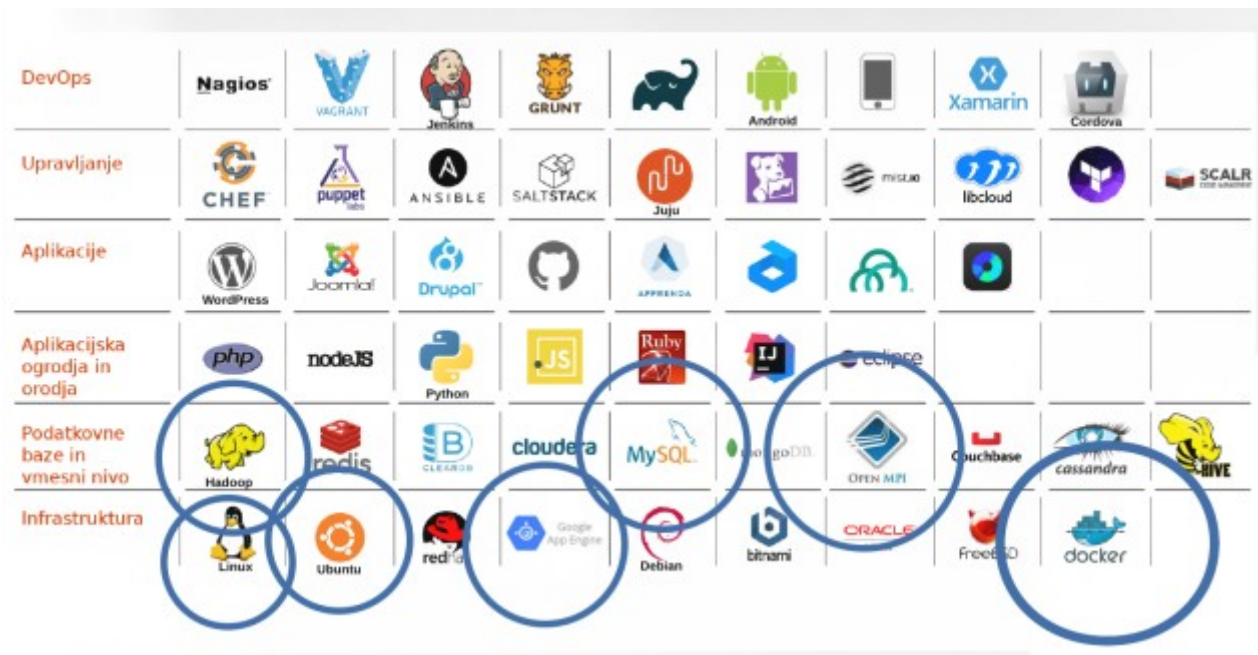
Watch a film, you need processes to pixel, manage...

Explain multiple graphical shells on a single Linux computer: With docker and virtualization, you can have more SO in your own computer.

Specify a command to change virtualbox environment settings: `vboxmanage`

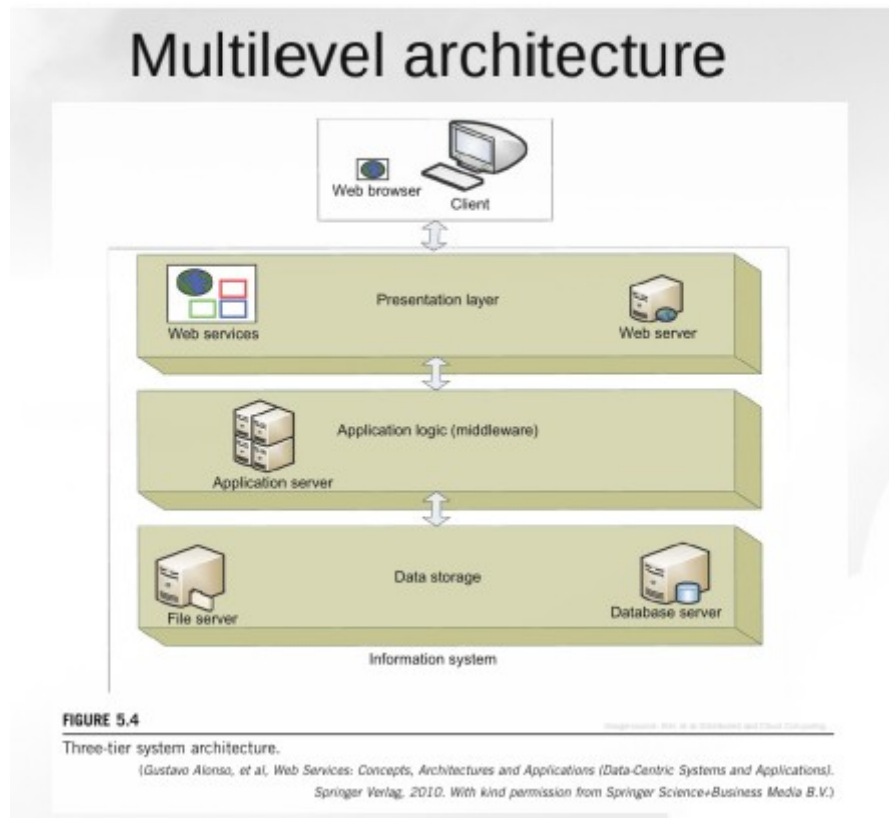
Give an example of how to build a cloud so that it allows multiple different concurrent environments: With more instances, for example, docker and them scalling mode.

Give some examples of cloud computing development programs and which sublayers they depend on:



2.9. Service Platform and Examples p.338

Give an example of building a platform using parts hosted in the cloud and what does it mean if it is **multi-level**: Multi-cloud is a strategy where an organization leverages two or more cloud computing platforms to perform various task



**What does the W3C offer as a platform through the definition of a service architecture?
Explain the purpose of REST and describe the components of a service-oriented SOA architecture**

The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) and serves as a forum for information, commerce, communication, and collective understanding.

REST (Representational state transfer) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web.

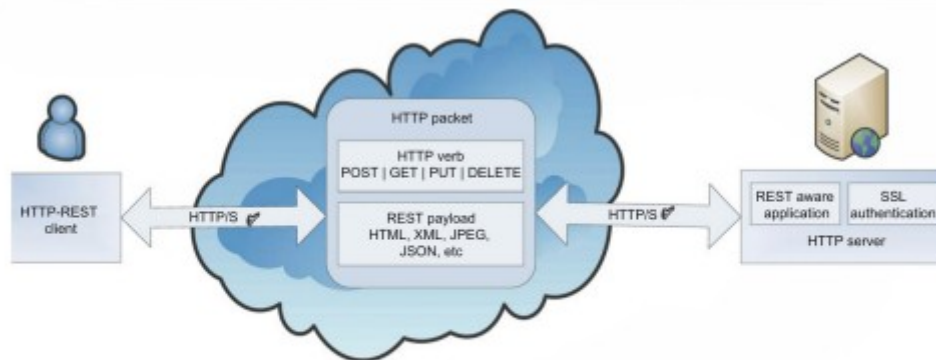


FIGURE 5.1

A simple REST interaction between user and server in HTTP specification.

(Courtesy of Thomas Fielding [2])

SOA (Service Oriented Architecture) defines a way to make software components reusable via service interfaces. These utilize common communication standards.

- Service provider: Security, availability...
- Service broker: Information for whoever ask for.
- Services requester: Locates entries.

Why do portals need to register and resolve resources if they run in the cloud?: Because instead of run in a cloud, they are split in different servers and in different levels.

What does the semantic web and service interaction programming contribute to building platforms through cloud computing?: Semantic web refers to the W3C vision of the web of linked data. This contribute to have more bases for create platforms.

Explain the roles of DNS and BGP in distributed service across a cloud platform: The DNS is for get the IP through a link, and Border Gateway Protocol (BGP) to advertise IP address ranges.

Which platform does Facebook use for the cloud? List some more services of AWS and Azure. Add a few more componentes for each of these platforms.

AWS: Can store databases, can offer servers, can offer space...

Model	IBM	Amazon	Google	Microsoft	Salesforce
PaaS	BlueCloud, WCA, RC2		App Engine (GAE)	Windows Azure	Force.com
IaaS	Ensembles	AWS		Windows Azure	
SaaS	Lotus Live		Gmail, Docs	.NET service, Dynamic CRM	Online CRM, Gifttag
Virtualization		OS and Xen	Application Container	OS level/ Hypel-V	
Service Offerings	SOA, B2, TSAM, RAD, Web 2.0	EC2, S3, SQS, SimpleDB	GFS, Chubby, BigTable, MapReduce	Live, SQL, Hotmail	Apex, visual force, record security
Security Features	WebSphere2 and PowerVM tuned for protection	PKI, VPN, EBS to recover from failure	Chubby locks for security enforcement	Replicated data, rule-based access control	Admin./record security, uses metadata API
User Interfaces		EC2 command-line tools	Web-based admin. console	Windows Azure portal	
Web API	Yes	Yes	Yes	Yes	Yes
Programming Support	AMI		Python	.NET Framework	

Note: WCA: WebSphere CloudBurst Appliance; RC2: Research Compute Cloud; RAD: Rational Application Developer; SOA: Service-Oriented Architecture; TSAM: Tivoli Service Automation Manager; EC2: Elastic Compute Cloud; S3: Simple Storage Service; SQS: Simple Queue Service; GAE: Google App Engine; AWS: Amazon Web Services; SQL: Structured Query Language; EBS: Elastic Block Store; CRM: Consumer Relationship Management.

Explain the purpose and operation of the InterCloud architecture: Intercloud or 'cloud of clouds' is a term refer to a theoretical model for cloud computing services based on the idea of combining many different individual clouds into one seamless mass in terms of on-demand operations.

There are 4 main types of cloud computing: private clouds, public clouds, hybrid clouds, and multiclouds. There are also 3 main types of cloud computing services: Infrastructure-as-a-Service (IaaS), Platforms-as-a-Service (PaaS), and Software-as-a-Service (SaaS):

- IaaS (Infrastructure as a Service): Gives service help. Ej: AWS, azure.
- PaaS (Platform as a Service): Center in the development of applications. Ej: Google App Engine.
- SaaS (Software as a Service): Is minimun. Ej. Office, dropbox...

How does distributed ledger provide a single cloud service? Through what do distributed ledger applications participate? Blockchains distribute transaction generation and confirmation (dapp)

2.10. Distribution and Clouds

Specify the types of cloud architectures and explain access to services and billing for them:
p361

- Private: Within the organization, more secure but more expensive
- Public: Registration (AWS)
- Community: Split costs
- Voluntary: P2P clouds

How do we send the load for Open MPI to the NorguGrid cloud using ARC tools

Describe the commands for compiling and running Open MPI programs. How do memory images of processes at different nodes differ? How do they receive parameters?

MPI work is divided into several processes and each node in more. For compile we use mipcc and for run mpirun. They receive parameters with messages.

Describe the MPI Communicator and range and how they are tied to distributed node collaboration. Also describe the syntax for sending and receiving base numbers

All the nodes are connected by MPI_Comm_rank() with the same program code but different rank. For messaging we use MPI_send() and MPI_receive()

Explain the difference between shared and shared memory and install OpenMP and OpenMPI. Also explain the types of OpenMPI calls for aggregating or processing node streams.

OpenMP enables distributed programming on extension and OpenMPI is directed towards distributed programming.

Why do we need to consider input and output delays in distributed parallelization? Because depending of this, it could change the order of the instructions and receive an error response.

2.11. Programming Examples

.-

2.12. Clouds and Assignment

What is the performance model and how do we use it to plan resource deployment for cloud deployment? Can we use resource elasticity in the cloud, what are assignment / forwarding / migration tasks?

What's the difference between load balancing with dynamic scheduling and big scheduling (bidding)? Bidding is used to select a vendor.

Which class do we use to set the sorter in MapReduce at Hadoop? Circular assignment sorting

List some algorithms for dynamically scheduling tasks Dynamic priority scheduling

2.13. Raft

What service does Raft provide? Why is it used in Docker?: Is an algorithm to manage the global cluster state. It helps in docker in swarm mode.

For Raft, explain the data for permanent or variable status (on leaders) and declaration of RPC request to vote on leader.

State	RequestVote RPC
Persistent state on all servers: (Updated on stable storage before responding to RPCs)	Invoked by candidates to gather votes (§5.2).
currentTerm latest term server has seen (initialized to 0 on first boot, increases monotonically)	Arguments:
votedFor candidateId that received vote in current term (or null if none)	term candidate's term
log[] log entries; each entry contains command for state machine, and term when entry was received by leader (first index is 1)	candidateId candidate requesting vote
Volatile state on all servers:	lastLogIndex index of candidate's last log entry (§5.4)
commitIndex index of highest log entry known to be committed (initialized to 0, increases monotonically)	lastLogTerm term of candidate's last log entry (§5.4)
lastApplied index of highest log entry applied to state machine (initialized to 0, increases monotonically)	Results:
Volatile state on leaders: (Reinitialized after election)	term currentTerm, for candidate to update itself
nextIndex[] for each server, index of the next log entry to send to that server (initialized to leader last log index + 1)	voteGranted true means candidate received vote
matchIndex[] for each server, index of highest log entry known to be replicated on server (initialized to 0, increases monotonically)	Receiver implementation:
	1. Reply false if term < currentTerm (§5.1)
	2. If votedFor is null or candidateId , and candidate's log is at least as up-to-date as receiver's log, grant vote (§5.2, §5.4)

Describe RPC AppendEntries for Raft:

AppendEntries and server rules

AppendEntries RPC

Invoked by leader to replicate log entries (§5.3); also used as heartbeat (§5.2).

Arguments:

term leader's term
leaderId so follower can redirect clients (to this leaderId)
prevLogIndex index of log entry immediately preceding new ones
prevLogTerm term of prevLogIndex entry
entries[] log entries to store (empty for heartbeat; may send more than one for efficiency)
leaderCommit leader's commitIndex

Results:

term currentTerm, for leader to update itself
success true if follower contained entry matching prevLogIndex and prevLogTerm

Receiver implementation:

1. Reply false if $\text{term} < \text{currentTerm}$ (§5.1)
2. Reply false if log doesn't contain an entry at prevLogIndex whose term matches prevLogTerm (§5.3)
3. If an existing entry conflicts with a new one (same index but different terms), delete the existing entry and all that follow it (§5.3)
4. Append any new entries not already in the log
5. If $\text{leaderCommit} > \text{commitIndex}$, set $\text{commitIndex} = \min(\text{leaderCommit}, \text{index of last new entry})$

Rules for Servers

All Servers:

- If $\text{commitIndex} > \text{lastApplied}$: increment lastApplied , apply $\text{log}[\text{lastApplied}]$ to state machine (§5.3)
- If RPC request or response contains term $T > \text{currentTerm}$: set $\text{currentTerm} = T$, convert to follower (§5.1)

Followers (§5.2):

- Respond to RPCs from candidates and leaders
- If election timeout elapses without receiving AppendEntries RPC from current leader or granting vote to candidate: convert to candidate

Candidates (§5.2):

- On conversion to candidate, start election:
 - Increment currentTerm
 - Vote for self
 - Reset election timer
 - Send RequestVote RPCs to all other servers
- If votes received from majority of servers: become leader
- If AppendEntries RPC received from new leader: convert to follower
- If election timeout elapses: start new election

Leaders:

- Upon election: send initial empty AppendEntries RPCs (heartbeat) to each server; repeat during idle periods to prevent election timeouts (§5.2)
- If command received from client: append entry to local log, respond after entry applied to state machine (§5.3)
- If last log index $\geq \text{nextIndex}$ for a follower: send AppendEntries RPC with log entries starting at nextIndex
 - If successful: update nextIndex and matchIndex for follower (§5.3)
 - If AppendEntries fails because of log inconsistency: decrement nextIndex and retry (§5.3)
- If there exists an N such that $N > \text{commitIndex}$, a majority of $\text{matchIndex}[i] \geq N$, and $\text{log}[N].\text{term} == \text{currentTerm}$: set $\text{commitIndex} = N$ (§5.3, §5.4).

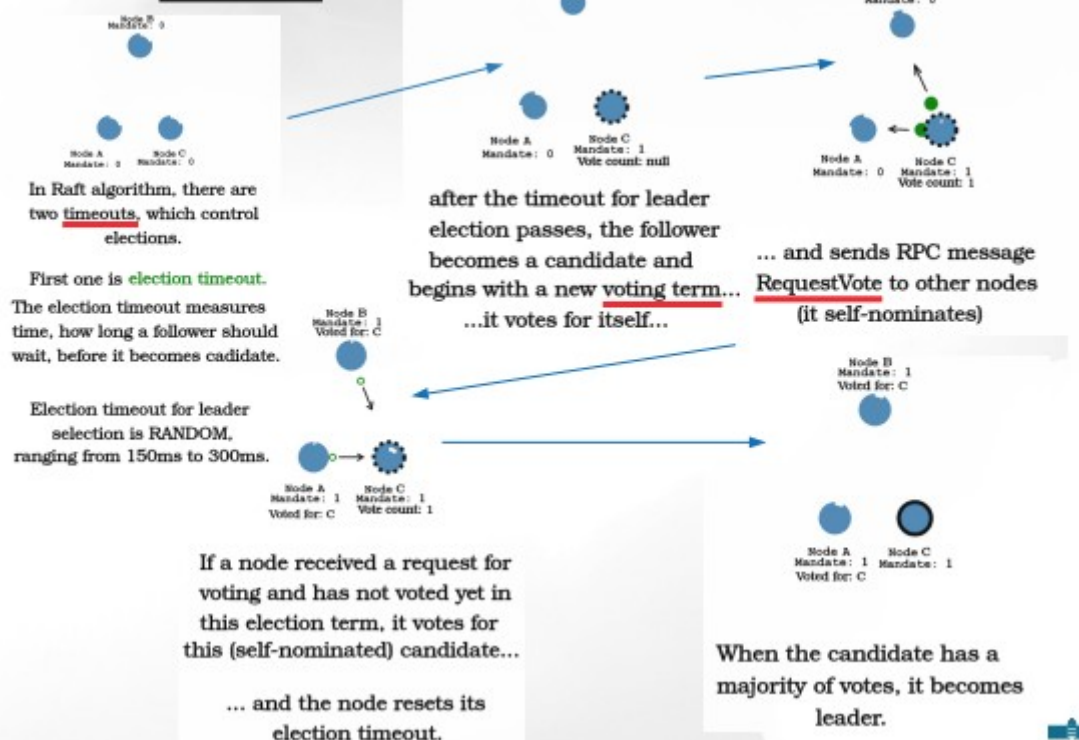
In the example, draw we need distributed consent for some data: Is a distributed consensus to choose a leader.

Draw an example of selecting a leader with three nodes where one of them fails. Which method is used to perform the heartbeat? What is an electoral mandate?

Explain the duplication of logs in Raft. What does the manager have to wait for before writing the entry?: When a new leader is elected we must replicated all interventions in the system by using the same RCP message type.

What are the time counters in Raft and how do they work? Draw another example of use in network failure

Raft: timers



2.14. Virtualization (general)

What are the three requirements must the virtual machine operator meet in terms of environment, speed and control?:

- Provides an environment equal to the basic environment.
- The speed of running programs in VMM is only slightly slower
- VMM has full control of the resources

What does network virtualization mean? What is data warehouse virtualization?: Means that one physical network can be split in one or more virtual networks, or the opposite.

Explain the ISA level of virtualization deployment: It has dynamic ISA (ISA emulation on MIPS). Binary instruction translation → Scan the instruction code and includes catching for common instructions, for other unchanged and upto 80-97% speed.

How do the levels of virtualization implementation differ, in what are the differences between the levels if virtualization with HAL, the runtime environment (OS) or libraries (e.g WINE, JVM...):

- HAL (Hardware Abstraction Layer): Xen hypervisor.

- OS level: Deployment of a Virtual Execution environment.
 - One VE, your own access to CPU, file system... but same OS kernel.
- Libraries: More virtualization level.

2.15 Questions from x86 virtualization to migration

2.16. Examples of Virtual Cloud Environments

Explain how to enclose virtual shells, users, and shell sessions in Linux. What does it mean to be able to run multiple graphical shells on one Linux Operating System at the same time?:

With the command `save` or `export` the .ova. It means to have an VMM architecture and more than one virtual machine running.

List some examples of hypervisors that allow you to create a virtual network: Hyper-V and VMWare. A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

2.17. MapReduce programming model

Explain the calculation model with MapReduce. Why don't we need locking for processing (reduce) with HDFS?

MapReduce is programming model Google for bigData, and where-in during runtime, system automatically parallelizes execution in the background.

Explain the example of assigning keys for word processing via MapReduce:

MapReduce programming model

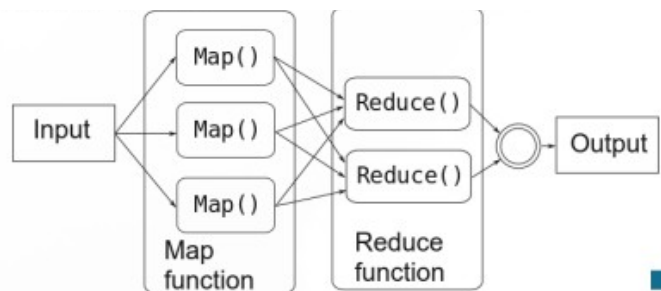
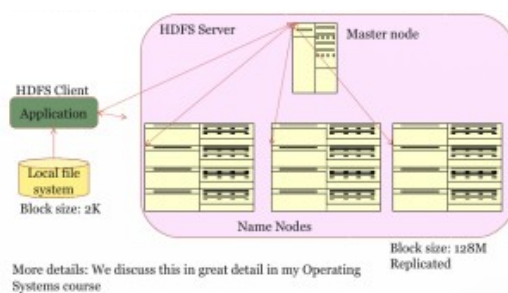
- Determine if the problem is parallelizable and solvable using MapReduce (ex: Is the data WORM?, large data set).
- Design and implement solution as Mapper classes and Reducer class.
- Compile the source code with hadoop core.
- Package the code as jar executable.
- Configure the application (job) as to the number of mappers and reducers (tasks), input and output streams
- Load the data (or use it on previously available data)
- Launch the job and monitor.
- Study the result.
- Detailed steps.

2.18. HDFS and GAE

Explain the architectural units of the HDFS service:

- NameNode: Contains metadata for HDFS.
- Secondary NameNode: Performs maintenance functions.
- DataNode: Keeps actual blocks of data HDFS.
- JobTracker: Manages tasks in MapReduce.
- TaskTracker: Supervises individual Map and Reduce tasks.

How do we schedule machining tasks in HDFS for data divided by MapReduce? Which HDFS servers are involved?



Explain data access using the GFS index model and Chubby lock:

2.19. Installing a Hadoop cluster in the cloud

Why we use /etc/hosts file and how is it related to DNS and cloud connections?: In Linux, /etc/hosts is a file used by the operating system to translate hostnames to IP-addresses. If we have a server, is like a DNS table.

What is an OpenSSH server and where do we use digital keys for it?: Is a server to connect two devices, one server and one client. The digital keys are to log in without password and verify that all is secure.

What the environmental variable HADOOP_HOME determines and what HADOOP_HDFS

Why do we use a connection to a network port in HDFS?: Because it has a lot of services

Specify some commands for shell work with HDFS:

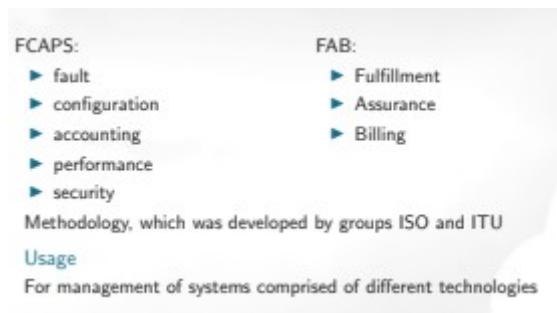


What does YARN care about? Which parameter specifies the class name for the sorter in ResourceManager?: YARN is responsible for allocating system resources to the various applications running in a Hadoop cluster and scheduling tasks to be executed on different cluster nodes. The parameter is “configuration”.

2.20. Hadoop word processing

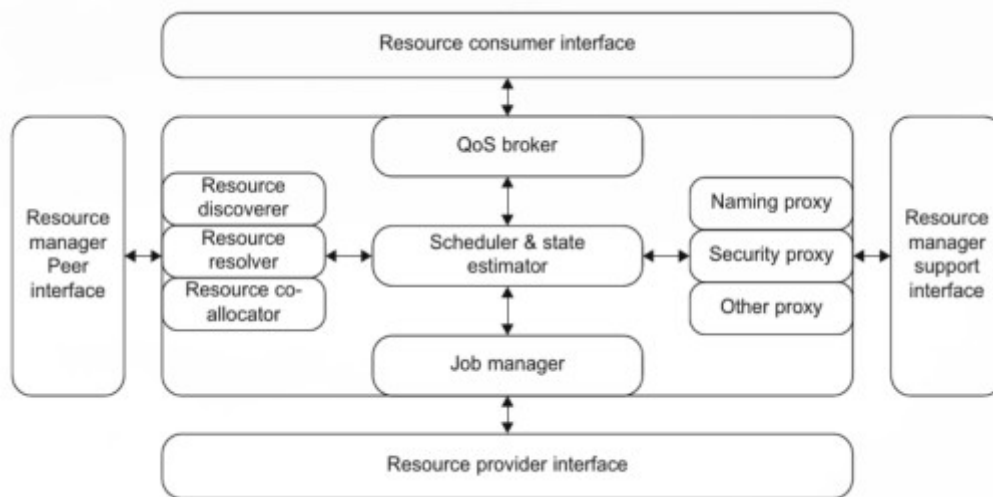
2.21. Cloud Monitoring and Management Tools

Describe FAB and FCAPS



What can Docker be used for in cloud monitoring and management?: Because we can save in images an entire Production environment.

What does the producter / consumer model provide and where is it used in Docker?

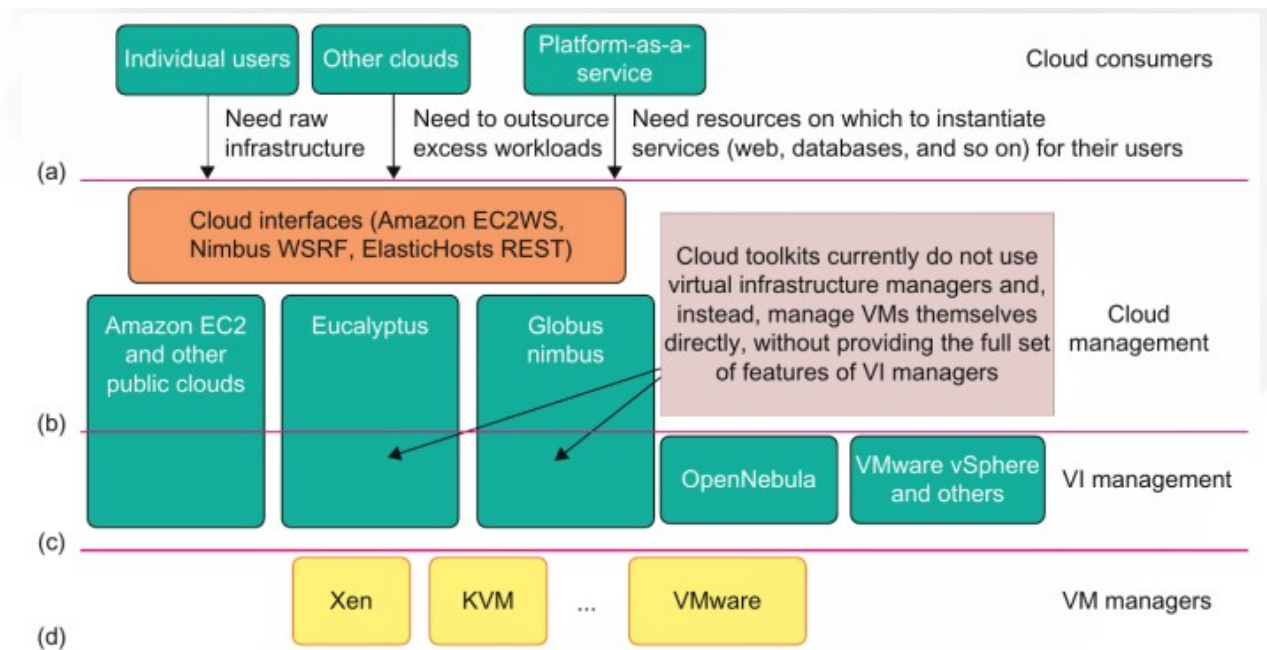


What is a resource explorer and how does it discover cloud compilation resources?:

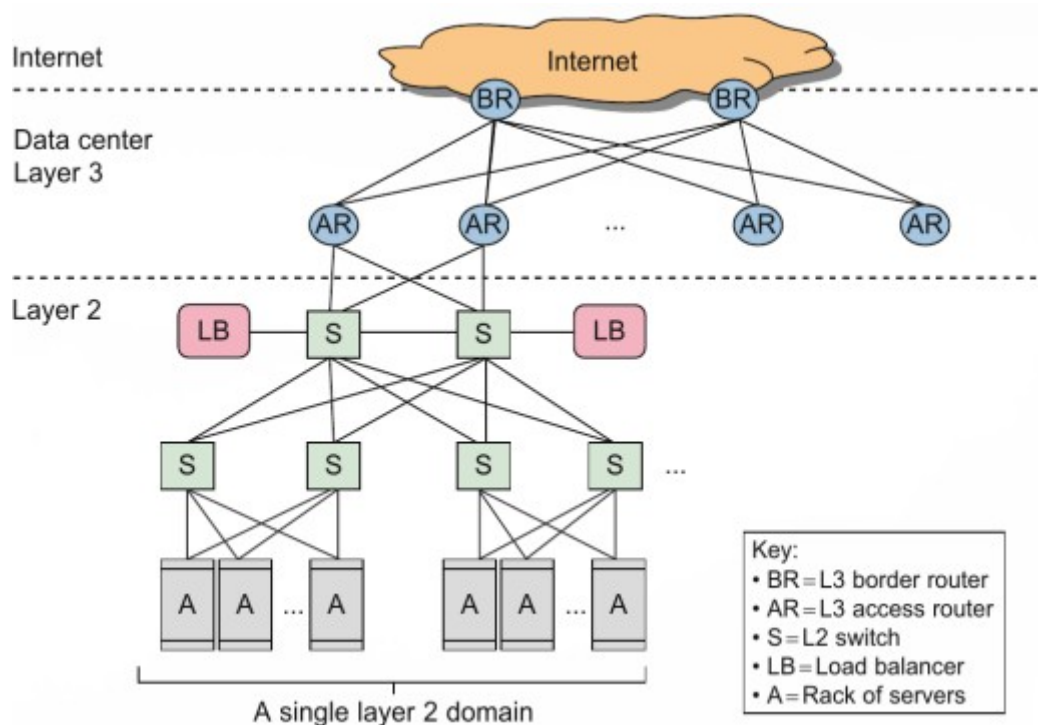
2.22: IaaS Hypervisor

Explain the difference in available resources between a system where Docker is run on a single laptop and a system where Docker is run on a cluster accessed via OpenSSH

Explain where the hypervisor appears in the cases of AWS, Azure and Google App Engine.



Explain the routing of network traffic within the data center architecture



Explain the difference between the architectural resources offered in the cloud and the service provided by the post-installed service.

Table 4.5 Five Major Cloud Platforms and Their Service Offerings [36]

Model	IBM	Amazon	Google	Microsoft	Salesforce
PaaS	BlueCloud, WCA, RC2		App Engine (GAE)	Windows Azure	Force.com
IaaS	Ensembles	AWS		Windows Azure	
SaaS	Lotus Live		Gmail, Docs	.NET service, Dynamic CRM	Online CRM, Gifttag
Virtualization		OS and Xen	Application Container	OS level/ Hypel-V	
Service Offerings	SOA, B2, TSAM, RAD, Web 2.0	EC2, S3, SQS, SimpleDB	GFS, Chubby, BigTable, MapReduce	Live, SQL Hotmail	Apex, visual force, record security
Security Features	WebSphere2 and PowerVM tuned for protection	PKI, VPN, EBS to recover from failure	Chubby locks for security enforcement	Replicated data, rule- based access control	Admin./record security, uses metadata API
User Interfaces		EC2 command-line tools	Web-based admin. console	Windows Azure portal	
Web API	Yes	Yes	Yes	Yes	Yes
Programming Support	AMI		Python	.NET Framework	

Note: WCA: WebSphere CloudBurst Appliance; RC2: Research Compute Cloud; RAD: Rational Application Developer; SOA: Service-Oriented Architecture; TSAM: Tivoli Service Automation Manager; EC2: Elastic Compute Cloud; S3: Simple Storage Service; SQS: Simple Queue Service; GAE: Google App Engine; AWS: Amazon Web Services; SQL: Structured Query Language; EBS: Elastic Block Store; CRM: Consumer Relationship Management.

Explain the difference between AWS and Microsoft Azure cloud. How is GAE different? Give a few more examples of the services offered in each:

- AWS:
 - Flexibility
 - Rent
 - Velocity
 - Elasticity
 - Security
- Azure:
 - For all ambits
 - IA security
 - Multi-cloud
 - Easier the development
- GAE:
 - Helper to create webs.