

## 2. Sistemas Distribuidos

Sistemas Distribuidos

Universidad Rey Juan Carlos

Juan Sebastián Cely Gutiérrez  
juan.cely@urjc.es



2024



(cc) 2024- GSYC

Algunos derechos reservados. Este trabajo se entrega bajo la licencia Creative Commons Reconocimiento - NoComercial - SinObraDerivada (by-nc-nd). Para obtener la licencia completa, véase <https://creativecommons.org/licenses/by-nc-nd/4.0/deed.es>

# Contenido

- 1 ¿Qué es un sistema operativo distribuido?
- 2 Arquitecturas paralelas
- 3 Virtualización
- 4 Computación en la nube
- 5 Referencias

## Section 1

¿Qué es un sistema operativo distribuido?

# OS distribuido

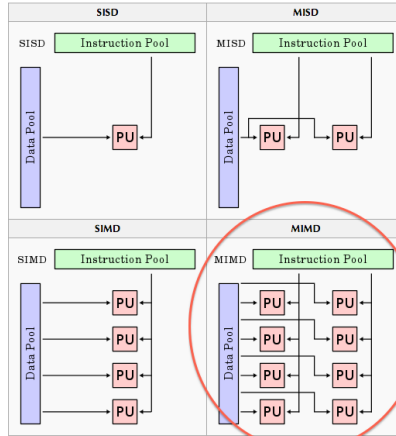
- OS distribuido: Los servicios del sistema ofrecidos por el OS están distribuidos (sistema de ficheros, procesos, autenticación etc.).
- Un OS distribuido... *¿Dónde?*

## Section 2

# Arquitecturas paralelas

# Taxonomía de Flynn

Una única instrucción sobre un único dato. Monoprocesador común.



Múltiples instrucciones sobre el mismo dato.  
Ejemplo: redundancia.

Múltiples datos para la misma instrucción  
Ejemplo: procesamiento vectorial.

Múltiples instrucciones sobre múltiples datos.

**SISTEMA DISTRIBUIDO**

# Modelos de programación paralela

- **Implícita:** el sistema (compilador, runtime y/o sistema operativo) se ocupan de paralelizar lo que se pueda. El programador se abstrae.
- **Memoria compartida:** distintos flujos de control concurrentes que acceden a una memoria común. El problema es la sincronización de los flujos de control en el acceso a la memoria.
- **Paso de mensajes:** se programa secuencialmente y los distintos componentes paralelos se comunican pasando mensajes entre ellos, pidiendose cosas. Problema: paso de estructuras de datos grandes. Si no hay **memoria compartida**, no se puede pasar un puntero.



# Multiprocesador

- Una máquina con varios procesadores y memoria compartida.
- Los procesadores están fuertemente acoplados.
- Distintas formas de conectar los procesadores con la memoria: bus, switched.

# UMA

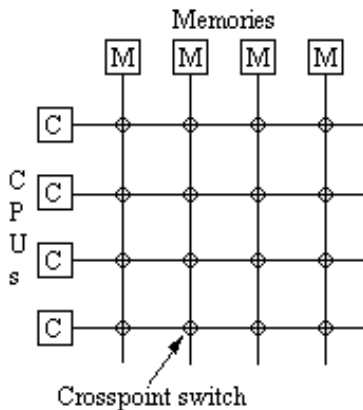
- Uniform Memory Access.
- Todos los procesadores están conectados a la memoria con un bus y acceden a la memoria de la misma forma (SMP).
- El bus es un cuello de botella.

# NUMA

- Non-Uniform Memory Access.
- Cada procesador (o conjunto de procesadores) está conectado a una memoria mediante un bus, a la que accede muy rapido.
- También puede acceder a las otras memorias, pero es más lento.
- Los procesadores están conectados en una red (p. ej. AMD hypertransport, Intel Quick-Path Interconnect)
- ccNUMA: Los distintos procesadores tienen caches de memoria y se garantiza la coherencia.
- nccNUMA: Las caches no mantienen coherencia. No encaja bien en el modelo de memoria compartida y ya apenas se usa.

## Ejemplo de conexión CPU-Memoria: *crossbar*

- Número de switches:  $N^2$



## Ejemplo de conexión CPU-Memoria: red *omega*

- Con  $N$  procesadores:
  - Número de etapas:  $\log_2(N)$
  - Número de switches por etapa:  $N/2$

**Ejemplo de rutado:** el procesador 100 quiere acceder a la memoria 010:

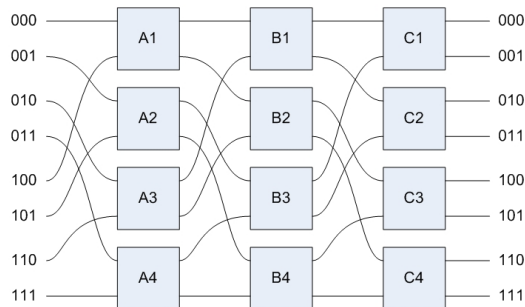
Etapa1: en A1 elige la salida 0  $\rightarrow$

B1

Etapa2: en B1 elige la salida 1  $\rightarrow$

C2

Etapa3: en C2 elige la salida 0  $\rightarrow$  memoria 010



## Ejemplo de Multiprocesador

Servidor supermicro AMD64 K10 (2012, aprox. 5000€):

- Multiprocesador con memoria compartida, fuertemente acoplado.
- Conexión de procesadores: switched
  - Hypertransport (6400 MT/s)<sup>1</sup>.
  - ccNUMA.
- Cache (línea de cache de 64 bytes):
  - L1 Datos (64 Kb), L1 Instrucciones (64 Kb), por core.
  - L2 Datos+Instrucciones (512 Kb), por core.
  - L3 (12 Mb), una para todos los cores.
- Protocolo de coherencia de caches: MOESI.
- Comunicación entre cores: IPIs.

---

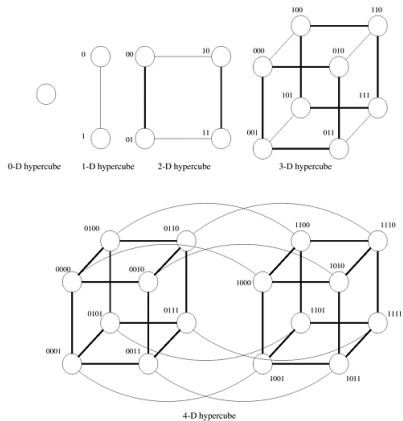
<sup>1</sup>Mega-transfers per sec.

# Multicomputador

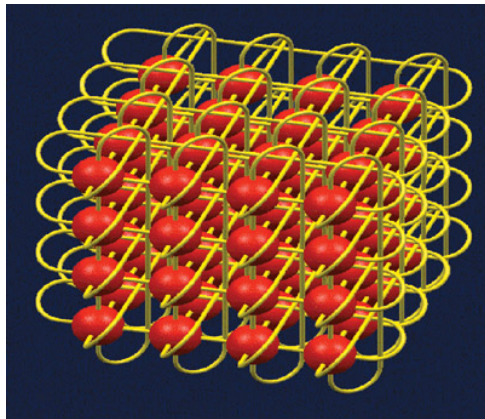
- Multicomputador  $\neq$  Multiprocesador
- Los procesadores son nodos conectados en red con una interfaz dedicada.
- Los procesadores están débilmente acoplados (depende de la red).
- Varias formas de interconectar los nodos: hypercube, torus, etc.

# Topologías

## Hipercubo:



## Toro:





## Ejemplo: Multicomputador

### IBM Blue Gene/Q (2012)

- Cada nodo es un multiprocesador con 16 CPUs 64-bit Power A2 con 16GB de RAM.
- Cada rack tiene 32 nodos.
- Total: 4096 nodos (65536 cores)
- Interconexión: torus de 5 dimensiones.

## Section 3

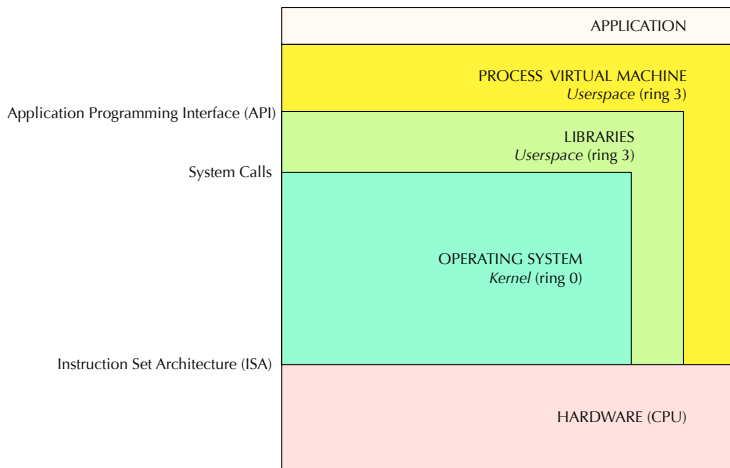
# Virtualización

# Máquinas Virtuales

Tipos:

- **Máquina virtual de proceso:** tiene como objetivo proporcionar una plataforma para ejecutar un único proceso.
- **Máquina virtual de sistema:** el VMM (VM Monitor) proporciona un entorno completo y persistente para ejecutar un OS y sus procesos.

# Máquinas Virtuales de Proceso



# Máquinas Virtuales de Proceso

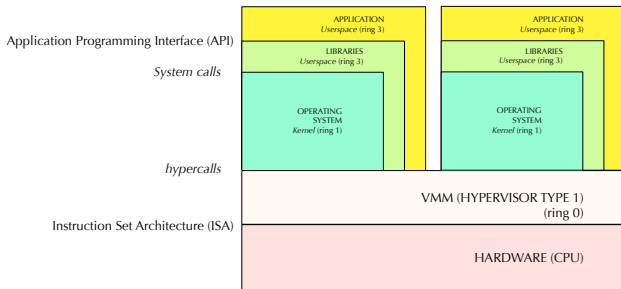
- **Multiprogrammed systems:** el propio OS!
- **Emuladores (dynamic binary translators):** ejecuta un programa cuya ISA es distinta a la de la máquina local. P. ej. OSX Rosetta.
- **Optimizadores:** traducen instrucciones de la misma ISA para mejorar la eficiencia. P. ej. Dynamo.
- **HLL VM:** ejecutan programas portables cuya ISA es virtual (bytecode). P. ej. .NET, Java.

# Máquinas Virtuales de Sistema

- VM clásica (hypervisor type 1 a.k.a. *bare metal* a.k.a. *unhosted*).
  - Paravirtualización.
  - Virtualización completa asistida por HW.
- VM alojada (hypervisor type 2).

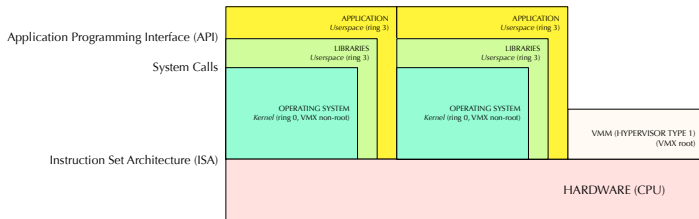
# Máquinas Virtuales de Sistema: paravirtualización

- El OS huésped está modificado para ejecutar sobre el VMM.
- El OS huésped realiza *hypercalls* para gestionar la tabla de páginas, planificar, poner timers, configurar el HW, etc.
- Ejemplos: Xen, KVM.



# Máquinas Virtuales de Sistema: asistidas por HW

- Se basa instrucciones especiales de la CPU para virtualización. P. ej. Intel VT-x, AMD-V.
- Instrucciones VMX: activar el modo VMX root, lanzar una VM, pasar el control al VMM, retomar una VM, etc.
- Además de ring 0-3, hay un modo especial en el que ejecuta el VMM: VMX root.
- El OS huésped no necesita modificaciones.
- Ejemplo: VMware vSphere.

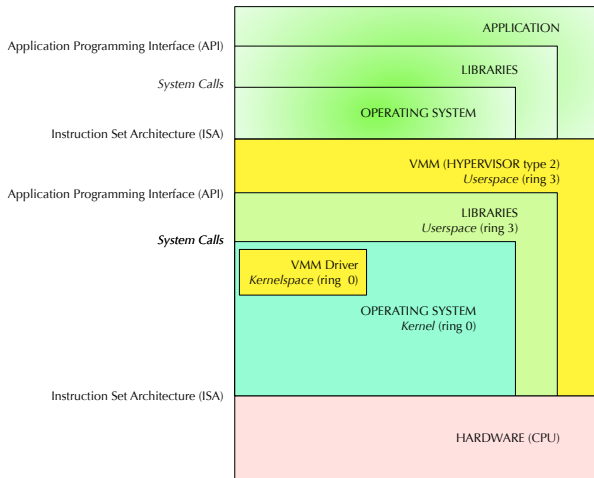




## Máquinas Virtuales de Sistema: alojada

- La VM se aloja sobre otro OS.
- El VMM puede instalar drivers en el OS anfitrión para mejorar el rendimiento. P. ej. VMWare Fusion, Virtual Box.
- *Whole-system* VM: la ISA de la VM no es la misma que la del HW y necesita **emulación**. P. ej. Virtual PC.

# Máquinas Virtuales de Sistema: alojada



## Virtualización a nivel de Sistema Operativo

- Una VM aísla distintas **imágenes completas** de distintos sistemas operativos ejecutando. Si lo que queremos aislar es un servicio, pagamos cierto coste ejecutando un OS completo para él (**tiempo en arrancar y parar la VM**, rendimiento, etc.).
- Solución: dentro del mismo sistema operativo (kernel) se pueden crear distintos entornos aislados, cada uno con sus propias abstracciones y recursos (espacio de procesos, sistema de ficheros raíz, CPU, recursos de red, usuarios, etc.).

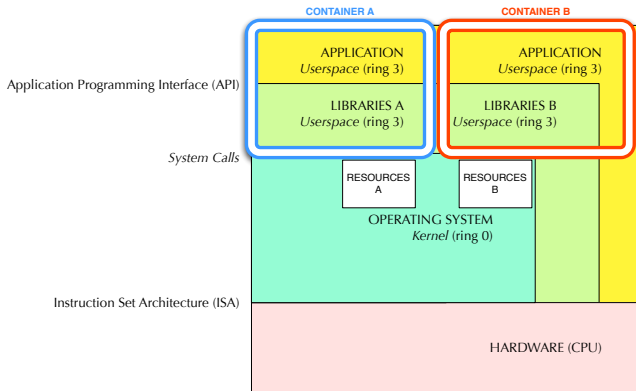
# Virtualización a nivel de Sistema Operativo

- El sistema operativo tiene que proporcionar mecanismos para aislar procesos. P. ej: Linux:
  - Linux Cgroups (control groups): permite crear grupos de procesos para controlar su acceso a dispositivos (/dev), limitar su uso de CPU, I/O, memoria...
  - Linux Namespaces: permite que cada proceso tenga su propia vista de: PIDs, red, IPC, FS.

# Virtualización a nivel de Sistema Operativo

- Pros: arranque rápido, más ligeros en general (mucha discusión, guerra abierta), no necesitas una imagen entera del sistema alojado.
- Contras: menos aislamiento, menos seguridad.
- Ejemplos: Docker, Linux Containers (LXC), OpenVZ, FreeBSD Jails, Solaris Zones, etc.

# Virtualización a nivel de Sistema Operativo: Contenedores



# Orquestación

- Usar **sistemas gestores de clusters** para orquestar distintos grupos de VMs/contenedores en un conjunto de máquinas físicas.
- Necesario internamente el los proveedores de Cloud.
- Ejemplos: VMware vRealize Orchestrator, Google Kubernetes, Docker Swarm, etc.

# Orquestación

## Ejemplo: Google Kubernetes

- Los componentes de tu aplicación/servicio distribuido ejecutan dentro de diferentes grupos de contenedores (*pod*) que comparten espacio de nombres (red, IPC, ...).
- Distintos *pods* forman *services* que tienen una dirección estática fija (ip, DNS) de cara al cliente.
- Kubernetes se encarga de replicar grupos de contenedores en el cluster, mantener vivas las réplicas (self-healing), balancear la carga distribuyendo las peticiones entre las réplicas...



## Section 4

# Computación en la nube

# Definición cloud computing

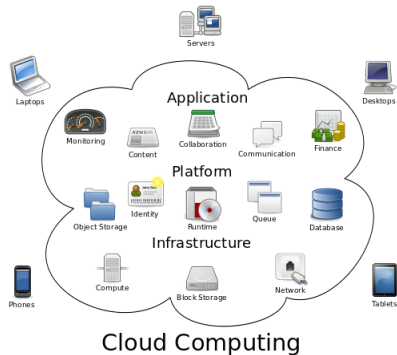
- El NIST define *cloud computing* como un modelo que posibilita acceso **ubícuo**, **cómodo** y **bajo demanda**, a través de una red de comunicación, a un conjunto de recursos de computación configurables.
  - Recursos tales como redes, servidores, almacenamiento, aplicaciones y servicios.
- Los recursos del cloud pueden **provisionarse rápidamente** y ser **liberados** con **mínimo esfuerzo de gestión** o intervención del proveedor de servicios.

## Definición cloud computing: conceptos clave

- Computación **bajo demanda**: Ya no es preciso una fuerte inversión inicial en costosos equipos físicos que luego hay que mantener y actualizar. Se paga por el acceso a recursos solo durante el tiempo que se utilizan.
- Computación **elástica**: Si existen picos de demanda de recursos de computación no es un problema. El sistema permite solicitar más recursos **dinámicamente**, y agregarlos a la infraestructura, así como **reasignar** recursos entre diferentes tareas y servicios.
- **Virtualización** de recursos: La tendencia general gira en torno a **encapsular el acceso** a cualquier recurso de computación físico (servidores, sistemas operativos, software, sistemas de conexión de redes, etc.) a través de objetos virtuales que ofrecen una API para interactuar con ellos.
  - Facilita el acceso del usuario a los recursos, ya que ofrece una capa uniforme de interacción, ocultando los detalles.
  - Facilita la gestión de recursos, al uniformizar los elementos de distinto tipo bajo una

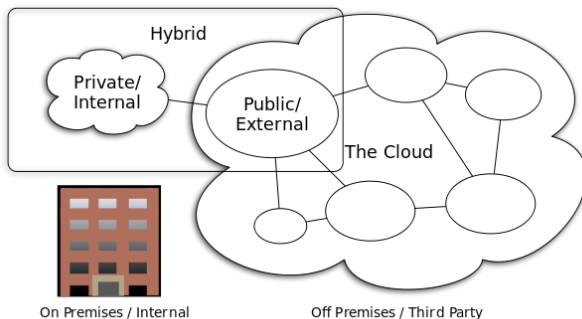
# Abstracción cloud computing

- La abstracción permite ocultar los elementos reales del proveedor de servicios a los usuarios, como si se tratase de una nube.



## Tipos de sistemas cloud

- Dependiendo de la gestión y el tipo de recursos en los que se basa, distinguimos tres tipos de sistemas cloud.



### Cloud Computing Types

CC-BY-SA 3.0 by Sam Johnston

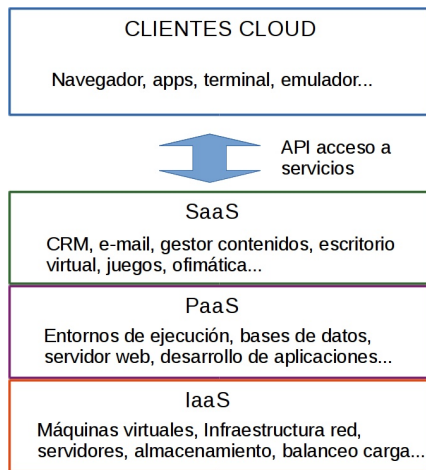
# Tipos de sistemas cloud

- Cloud **privada**: El acceso a los recursos en la nube es exclusivo del propietario de los servicios y se realiza mediante una conexión privada.
- La infraestructura física de la nube suele estar operada de forma privada por una sola organización, independientemente de que la gestión sea interna o a través de un proveedor externo.
- En cuanto a su ubicación, puede ser estar tanto en instalaciones de la propia compañía como en un proveedor externo, pero siempre que proporcione los recursos en exclusiva.
  - Algunos proveedores de servicios de cloud pública como Amazon AWS o Microsoft Azure pueden ofrecer también acceso exclusivo y seguro a los servicios, con un coste adicional.

## Tipos de sistemas cloud

- Cloud **pública**: El acceso a los recursos en la nube no es exclusivo, sino que se ofrece un servicio público para cualquier usuario al que se puede acceder a través de Internet. Tanto la infraestructura física como la ubicación pueden ser, como en el caso anterior, internas u ofrecidas mediante un proveedor, pero en esta modalidad el acceso a los recursos no es exclusivo, lo que implica importantes diferencias de seguridad.
- Cloud **híbrida**: En una solución compuesta por recursos de las dos modalidades anteriores, en la que el cliente del servicio puede elegir dónde despliega sus servicios o aplicaciones en función de diversos requisitos como coste o medidas de seguridad necesarias.

# Modelos de servicio en cloud





## Modelos de servicio cloud: IaaS

- Nivel de servicio más básico que se ofrece sobre cloud.
- El cliente tiene acceso directo a recursos que son **abstracciones de la infraestructura** real que aprovisiona el cloud.
- Típicamente incluye servicios tales como: virtualización, funciones y servicios de red virtuales (SDN y NFV), almacenamiento en crudo, de archivos y objetos serializados, filtrado de tráfico, balanceo de carga, etc.
- Elimina labores de mantenimiento y actualización de infraestructura física (consumo energético, actualización hardware, etc.).
- Sigue precisando conocimientos sobre administración y gestión de los recursos utilizados.

## Modelos de servicio cloud: PaaS

- Típicamente ofrecidos para desarrollo de aplicaciones.
- El servicio oferta un entorno de trabajo preparado para desarrollo software en una plataforma específica: lenguajes y sistemas de computación (.NET, Java, Python, Ruby, servicios Google, etc.).
- El desarrollador puede centrarse en la creación de aplicaciones, liberándose de las tareas de gestión y mantenimiento de la plataforma computacional subyacente.
- Un caso importante es el de **data Platform as a Service (dPaaS)**, donde el proveedor ofrece una plataforma vertical integrada para análisis de datos e integración con servicios basados en datos.
  - Ej: IBM Bluemix con Spark.

## Modelos de servicio cloud: SaaS

- El usuario obtiene acceso a aplicaciones de alto nivel, dirigidas a usuario/cliente final.
- Suele requerir el pago por tiempo de uso o por suscripción.
  - Ejemplo: Microsoft Office 365.
- El usuario ya solo se ocupa de interactuar con el software, todos los demás detalles sobre plataforma e infraestructura son transparentes.
- El proveedor de servicios tiene control total sobre la versión de software utilizado, acceso a recursos, etc.

## Ejemplo: Databricks

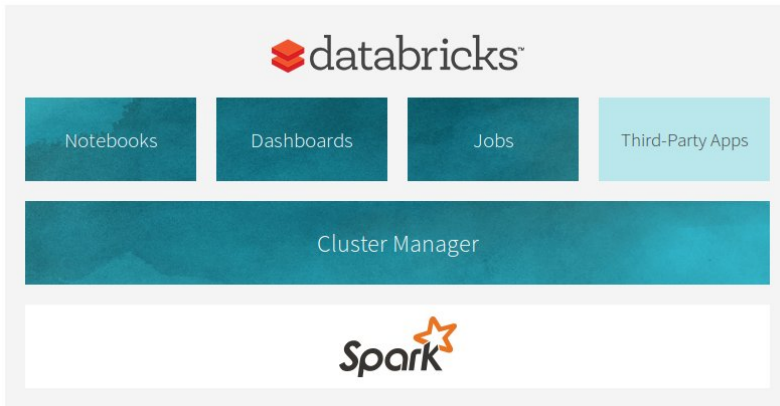


Figura: Arquitectura del servicio Databricks

## Ejemplo: databricks

- Producto proporcionado por la empresa del mismo nombre, que ofrece un servicio de dPaaS.
- El cliente tiene acceso a varios componentes:
  - **Notebooks** compatibles con Jupyter, para implementar procesos de análisis de datos y acelerar su replicación y mejora, compartiéndolos con otros usuarios.
  - **Gestión de plataforma** para controlar la creación asistida de clústers, accediendo a la cuenta del cliente para infraestructura con un proveedor IaaS (actualmente, solo compatible con Amazon AWS).
  - Gestor de **trabajos**, que permite monitorizar la ejecución de tareas de obtención, procesamiento y almacenamiento de datos en el clúster.
  - **Dashboards** para la creación de informes visuales a partir de resultados de análisis de datos generados mediante Spark.
  - Conectores con **aplicaciones de otros proveedores**: Pentaho, Qlik, TIBCO Spotfire, Tableau, Pantera o Zoomdata.

# Ejemplo: Databricks

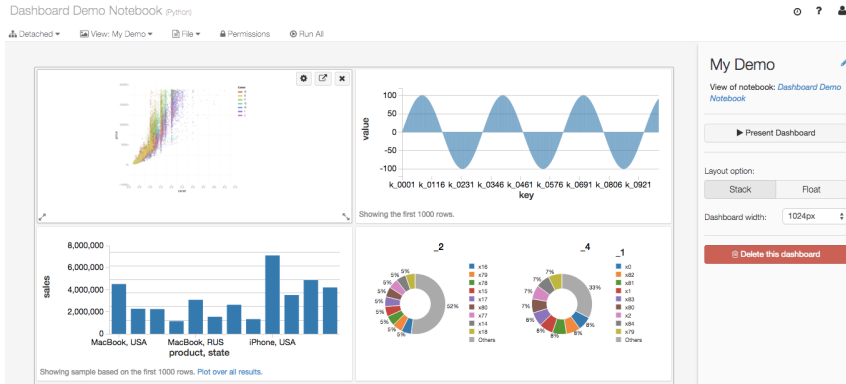



Figura: Ejemplo de dashboard en Databricks

# Bibliografía I



- *Mark D. Hill, Norman P. Jouppi, and Gurindar S. Sohi.* Chapter 9 "Multiprocessors and Multicomputers," from Readings in Computer Architecture, Morgan Kaufmann, 2000
- *A. S. Tanenbaum.* Operating Systems, design and implementaiton. Pearson Prentice Hill.
- *A. S. Tanenbaum.* Distributed Systems. Pearson Prentice Hill.
- *A. S. Tanenbaum.* Modern Operating Systems. Pearson Prentice Hill.
- *A. Silberschatz.* Operating Systems. Wiley.
- *J. E. Smith and R. Nair.* The Architecture of Virtual Machines. IEEE Computer 38, 5. 2005.

## Bibliografía II

- *D. Bernstein. Containers and Cloud: From LXC to Docker to Kubernetes. Cloud Computing, IEEE (Volume:1 , Issue: 3 ).*
  - *A. Madhavapeddy and D. J. Scott. Unikernels: Rise of the Virtual Library Operating System. ACM Queue. Volume 11, issue 11.*
  - *Blue Gene/Q Overview and Update*  
[https://www.alcf.anl.gov/files/IBM\\_BGQ\\_Architecture\\_0.pdf](https://www.alcf.anl.gov/files/IBM_BGQ_Architecture_0.pdf)
-  [van Steen & Tanenbaum, 2017] van Steen, M., Tanenbaum, A. S.  
*Distributed Systems.*  
Third Edition, version 01. 2017.



## Bibliografía III

-  [Colouris et al., 2011] Colouris, G., Dollimore, J., Kindberg, T., Blair, G.  
*Distributed Systems. Concepts and Design*.  
Pearson, May, 2011.
-  [Ortega et al., 2005] Ortega, J., Anguita, M., Prieto, A.  
*Arquitectura de Computadores*.  
Ediciones Paraninfo, S.A. 2005.



Juan Sebastián Cely Gutiérrez

Intelligent Robotics Labs.

[juan.cely@urjc.es](mailto:juan.cely@urjc.es)



2024