Arquitectura de software en la nube

Conceptos básicos

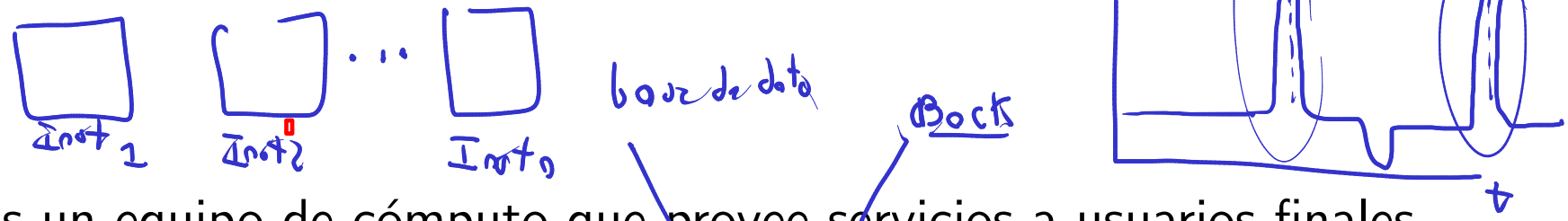
Carlos Andrés Delgado Saavedra, Msc

Universidad San Buenaventura Cali

Julio de 2024

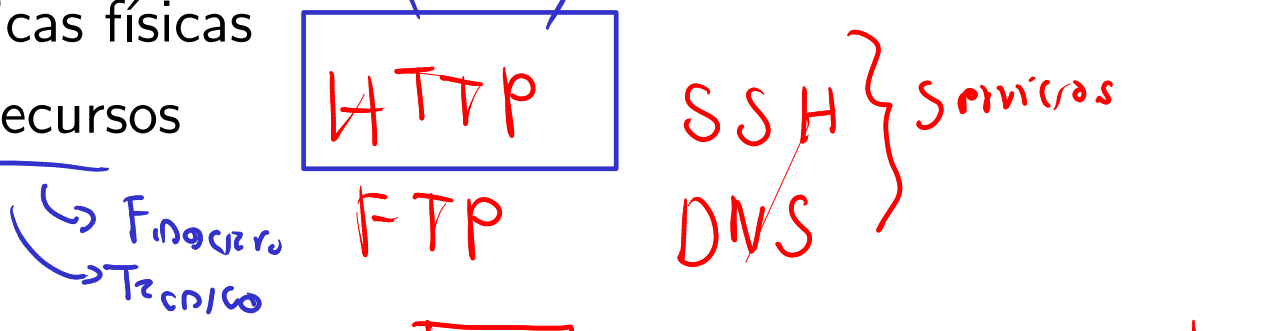
Contenido

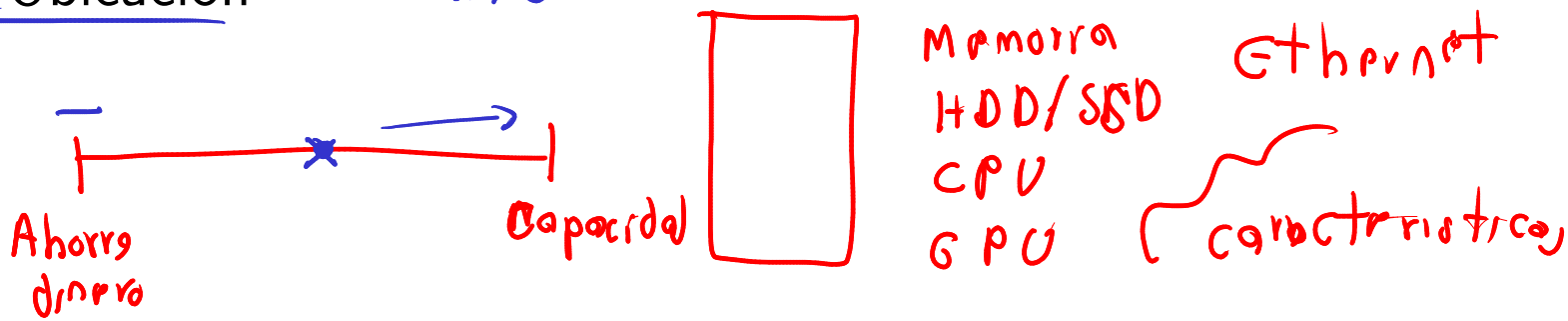
Servidores



Es un equipo de cómputo que provee servicios a usuarios finales.

- Características físicas
- Gestor de recursos
- Software
- Ubicación





Almacenamiento

Nos provee persistencia en los datos que maneja una aplicación

- En bloques: Se utiliza un identificador para acceder a las partes de un dato
  - En archivos: Estructura jerárquica de datos
- En objetos: imágenes, documentos, sitios estáticos

Almacenamiento por bloques

- Acceso directo a los bloques de datos
- Adecuado para sistemas tradicionales
- Caso 1: Bases de datos relacionales
- Caso 2: Máquinas virtuales (IaaS) ← Virtualbox Docker

Almacenamiento por archivos

- Estructura jerárquica
- Gestión de permisos
- Caso 1: Repositorios de acceso compartido
- Caso 2: Sistema de archivos distribuidos

Almacenamiento por objetos

- Acceso a través de identificadores URL
- Escalabilidad horizontal
- Caso 1: Almacenamiento de datos no estructurados
- Caso 2: Servicios de almacenamiento en la nube

Bases de datos I

1. Base de datos de series temporales
   - Almacenamiento datos de tiempo
   - Consultas rápidas sobre grandes volúmenes de datos
   - Integración con herramientas de visualización de datos

   Ejemplos: Time Stream, Google BigTable, Azure Time Series Insights

Bases de datos II

2. Base de datos de documentos
   - Modelado de datos
   - Consultas flexibles sobre los datos
   - Soporte transaccional ACID (Atomocidad, Consistencia, Aislamiento, Durabilidad)

   DocumentDB, Google Cloud Firestore, Amazon CosmosDB

Bases de datos III

3. Base de datos relacionales:
   - Escalabilidad vertical y horizontal
   - Soporte ACID
   - Optimización de Consultas
   - Enfoque transaccional
   - Compatibilidad extendida

   Google Cloud SQL, Azure SQL Database, Amazon RDS

Bases de datos IV

4. Base de datos no SQL:
   - Escalabilidad horizontal
   - Alta disponibilidad
   - Optimización para ciertas consultas
   - Modelo de datos flexible

   Google Cloud Datastore, Azure Table Storage, Amazon CosmosDB

Bases de datos V

5. Bases de datos en memoria:
   - Velocidad de acceso
   - Bajo latencia
   - Escalabilidad horizontal y vertical
   - Alta disponibilidad

   Amazon ElastiCache, Google Cloud Memorystore, Azure Cache for Redis

Bases de datos VI

6. Bases de datos clave-valor:
   - Escalabilidad horizontal
   - Alta disponibilidad
   - Bajo costo de almacenamiento
   - Modelo de datos simple

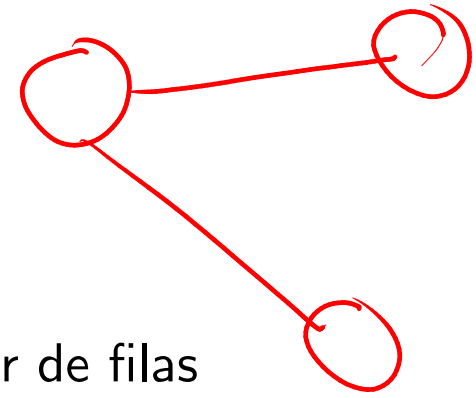
   Azure CosmosDB (modo clave-valor), Firestore, Amazon DynamoDB (modo clave-valor)

Bases de datos VII

7. Bases de datos orientadas a grafos:
   - Modelado de datos como entidades y relaciones
   - Consultas complejas para datos conectados
   - Soporte para algoritmos de grafos
   - Capacidad para representar datos complejos

   Google Cloud Datastore, Azure Table Storage, Amazon Neptune

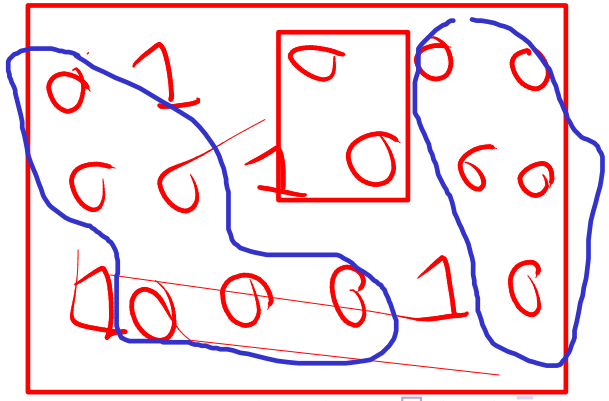
Bases de datos VIII



8. Bases de datos orientadas a columnas:
   - Almacenamiento en columnas en lugar de filas
   - Consultas rápidas
   - Compresión de datos
   - Escalabilidad horizontal

   Google BigTable, Azure Synapse Analytics, Amazon Redshift (PostgreSQL)

Vectores/Matrices dispersas



networkx <-- Python

Infraestructura como código (IAC)

- Automatización de la infraestructura
- Despliegue de aplicaciones
- Configuración de servidores
- Escalabilidad
- Descripción de infraestructura en lenguajes basados en YAML, JSON y XML

Terraform, Ansible, Chef, Puppet, Vagrant

Funciones como servicio (FaaS)

- Ejecución de código
- Escalabilidad automática
- Pago por uso
- Sin servidor (Severless)
- Eventos como disparadores

Limitados

PAAS

AWS Lambda, Google Cloud Functions, Azure Functions

Contenedores

Unidad de software que **encapsula** una aplicación y sus dependencias en un entorno aislado del sistema operativo.

- Portabilidad
- Escalabilidad
- Eficiencia
- Orquestación

Paravirtualización <-- S.O host

Virtualización <-- Emulación

Docker, Kubernetes, OpenShift

Microservicios

- Tolerancia a fallos
- Desarrollo ágil de aplicaciones
- Escalabilidad
- Mantenimiento
- Despliegue

Serialización de datos

- Formato de intercambio de datos
- Representación de datos interpretable
- Comunicación entre sistemas
- Enfocada en la eficiencia de construir (serializar) y reconstruir (deserializar) datos

XML, JSON, YAML, Protocol Buffers, Avro

Ejemplo de serialización en JSON

```
{
  "nombre": "Carlos",
  "edad": 30,
  "ciudad": "Cali",
  "habilidades": {
    "lenguajes": ["Python", "Java", "
       JavaScript"],
    "bases_de_datos": ["PostgreSQL", "
       MongoDB"]
  }
}
```

Ejemplo de serialización XML

```
<persona>
  <nombre>Carlos</nombre>
  <edad>30</edad>
  <ciudad>Cali</ciudad>
  <habilidades>
  <lenguajes>
    <lenguaje>Python</lenguaje>
    <lenguaje>Java</lenguaje>
    <lenguaje>JavaScript</lenguaje>
  </lenguajes>
  <bases_de_datos>
    <base_de_datos>PostgreSQL</base_de_datos>
    <base_de_datos>MongoDB</base_de_datos>
  </bases_de_datos>
  </habilidades>
</persona>
```

Ejemplo de serialización YAML

```yaml
nombre: Carlos
edad: 30
ciudad: Cali
habilidades:
  lenguajes:
    - Python
    - Java
    - JavaScript
  bases_de_datos:
    - PostgreSQL
    - MongoDB
```

On-premises

- Infraestructura local: Ubicación física
- Control total
- Seguridad
- Costos fijos
- Escalabilidad limitada

¿Siempre será más costoso que la nube?

Contenido

Cloud native

- Desarrollo de aplicaciones en la nube
- Independiente del sistema operativo
- Despligue continuio
- Escalabilidad automática
- Predecible y mutable

Cloud native foundation (CNF)

- Organización sin fines de lucro
- Estándares y buenas prácticas
- Proyectos de código abierto
- Certificaciones

Herramientas: Argo, Helm: Despliegue de aplicaciones.
Prometheus, fluentd: Monitoreo de aplicaciones. etcd, Kubernetes: Orquestación de contenedores, 0Auth, Keycloak: Seguridad

Beneficios de cloud native

- Escalabilidad
- Arquitectura modular
- Eficiencia
- Seguridad
- Despliegue continuo
- Costos

Contenido

Características de Severless

- Sin servidor
- Escalabilidad automática
- Pago por uso
- Eventos como disparadores
- Eficiencia

AWS Lambda, Google Cloud Functions, Azure Functions

Retos serverless

- Tiempo de inicio
- Tiempo de ejecución
- Límites de recursos
- Dependencias
- Monitoreo

Componentes

- Streams o flujos
- Colas
- Datastore: no SQL
- Bucket: Almacenamiento objetos
- API
- Identity and Access Management (IAM)

Ejemplo Azure Functions

```python
import logging
import azure.functions as func
import json

def main(req: func.HttpRequest) -> func.HttpResponse:
    logging.info('Python HTTP trigger function processed a
        request.')
    data = {
        "nombre": "Carlos",
        "edad": 30,
        "ciudad": "Cali",
        "habilidades": {
            "lenguajes": ["Python", "Java", "JavaScript"],
            "bases_de_datos": ["PostgreSQL", "MongoDB"]
        }
    }
    return func.HttpResponse(json.dumps(data))
```

Contenido

Características

- Redundancia: Componentes duplicados, diferentes localizaciones y zonas de disponibilidad
- Escalabilidad automática: Picos de tŕafico
- Monitoreo y recuperación automática: Detección de problemas y procedimientos
- Balanceo de carga: Distribuir el tráfico en diferentes instancias
- Pruebas continuas: Asegurar eficiencia y calidad

Tolerancia a fallos

- Detección de fallos
- Recuperación de fallos
- Mitigación de fallos
- Resiliencia
- Pruabas de fallos

Escalabilidad

1. Escalabilidad horizontal
   - Aumentar capacidad: CPU, memoria, almacenamiento
   - Aplicaciones de alto rendimiento
2. Escalabilidad horizontal
   - Incrementar capacidad al distribuir carga de trabajo
   - Aplicaciones que pueden dividir el trabajo en tareas independientes