UNIVERSIDAD DE GRANADA

DECSAI

Master Profesionalizante en Ingeniería Informática

Cloud Computing: Servicios y Aplicaciones

UNIVERSIDAD DE GRANADA

DECSAI

T3. Otros servicios en Cloud Computing

Contenido

- X as a Service
- Data as a Service
- Machine Learning as a Service
- Identification as a Service
- Storage as a Service
- Function as a Service
- Otros servicios

X as a Service

Everything as a Service

- Servicios gestionados por el proveedor: el cliente no se encarga de la infraestructura ni gestión de la aplicación
- Usos a través de:
  - Aplicación web
  - Aplicaciones nativas
  - APIs

- Las nuevas *start-ups* suelen desplegar sus productos como SaaS
- El listado de servicios está en plena expansión

Data as a Service

Data as a Service (DaaS)

- Hace referencia a un SGBD ejecutándose sobre una plataforma cloud
- El servicio ofrece automáticamente y de forma transparente gestión, **escalabilidad** y **alta disponibilidad**.
- Acceso al servicio a través de API y consola web

RDBMS en cloud

- Amazon Relational Database Service (RDS): https://aws.amazon.com/es/rds/
  - Motores: Aurora, PostgreSQL, MySQL, MariaDB, Oracle, SQL Server
- Azure SQL Database: http://azure.microsoft.com
- Oracle Database: http://cloud.oracle.com

Aplicaciones empresariales

Computación

Interacción con clientes

Base de datos

Herramientas para desarrolladores

Amazon Aurora
Base de datos relacional administrada de alto rendimiento

Amazon DynamoDB
Base de datos NoSQL administrada

Amazon DocumentDB (compatible con MongoDB)
Base de datos de documentos completamente administrada

Amazon ElastiCache
Sistema de almacenamiento de caché en memoria

Servicio Apache Cassandra administrado por Amazon
Base de datos administrada compatible con Cassandra

Amazon Neptune
Servicio completamente administrado de base de datos de gráficos

Amazon Quantum Ledger Database (QLDB)
Base de datos de libro mayor completamente administrada

Amazon RDS
Servicio administrado de bases de datos relacionales para MySQL, PostgreSQL, Oracle, SQL Server y MariaDB

Amazon RDS on VMware
Automatice la administración de bases de datos locales

Amazon Redshift
Almacenamiento de datos rápido, sencillo y rentable

Amazon Timestream
Base de datos de serie temporal completamente administrada

AWS Database Migration Service
Migre bases de datos con tiempo de inactividad mínimo

Clasificación SGBD

- Bases de datos SQL: servicio para SGBD tradicionales (relacionales)
  - Las restricciones que implementan permiten consultas (en SQL) complejas
  - No escalan bien (eficientemente)
- Bases de datos No SQL

SGBD NoSQL

- NoSQL (not only SQL) es una categoría general de sistemas de gestión de bases de datos que difiere de RDBMS: no hay esquemas, no permiten JOINs, no intentan garantizar ACID y escalan horizontalmente
- Son almacenamiento estructurado (no necesariamente tablas)

Objetivos de BD NoSQL

- Distribución
- Escalabilidad: horizontal y prácticamente ilimitada
- Disponibilidad: mediante distribución y replicación
- Tolerancia a fallos
- Flexibilidad: Modificación del esquema e incorporación de nuevos tipos de datos posteriormente a la creación de la BD

Almacenamiento

- A diferencia de RDBMS no se define por adelantado el esquema para incorporar información real
- En BD distribuidas:
  - Almacenes basados en filas y columnas
  - Consistencia eventual
  - *Sharding*
  - Replicación maestro-maestro
  - Particionado

ACID vs. BASE

- En los RDBMS las transacciones **ACID** (Atomicidad, Consistencia, Aislamiento y Durabilidad) garantizan la consistencia y estabilidad de las operaciones → Gestión de bloqueo
  - **Atomicidad**: garantía de ejecución o no de la operación
  - **Consistencia**: integridad
  - **Aislamiento**: ejecución concurrente coherente con ejecución secuencial
  - **Durabilidad**: persistencia

ACID vs. BASE

- NoSQL, almacenamiento según modelo BASE:
  - Basic availability
  - Soft-state
  - Eventual consistency
- BASE: alternativa flexible a ACID para cuando no se requiere un modelo estricto según los requisitos del modelo relacional

Características para NoSQL

- Garantías de consistencia débiles
- Arquitectura distribuida, elementos redundantes
- Estructuras de datos sencillas: arrays asociativos o almacenes para pares clave-valor

Características para NoSQL

- Fáciles de usar en clusters
- Guardan datos persistentes
- No hay esquemas fijos, sino dinámicos
- Sistemas de consultas propios
- Propiedades ACID en un nodo del cluster

¿Por qué son necesarias?

- Desafíos presentados por las aplicaciones web modernas, computación ubicua y Big Data (IoT)
  - Datos a escala web
  - Procesamiento masivo de datos
  - *Alta frecuencia* de lecturas y escrituras
  - Las aplicaciones sociales (no bancarias) no necesitan el mismo nivel de ACID

¿Por qué son necesarias?

- Desafíos:
  - RDBMS no escalan con el tráfico a un coste aceptable
  - El tamaño del esquema crece desproporcionadamente.
  - Muchos datos temporales
  - La BD se desnormaliza por rendimiento o conveniencia
  - Consultas sobre relaciones jerárquicas complejas; recomendaciones o inteligencia de negocio
  - Transacciones locales no muy durables

Tipos de BDDD

- Orientadas a columnas
- Orientadas a documentos
- De clave-valor
- Basadas en grafos

Ejemplos de DBMS No SQL

- MongoDB: https://www.mongodb.com
  - Orientada a documentos
- Cassandra: http://cassandra.apache.org
  - Tipo clave-valor
- Redis: https://redis.io
  - Tipo clave-valor
- CouchDB: http://couchdb.apache.org
  - Orientada a documentos
- Titan: http://titan.thinkaurelius.com
  - Base de datos de grafos

Ejemplos de DBMS No SQL (2)

- HBase: https://hbase.apache.org
  - Estructura tabular
- BigTable:
  - Estructura tabular
- ToroDB: https://github.com/torodb/stampede/
  - Híbrido entre MongoDB y PostgreSQL
- Amazon Neptune: https://aws.amazon.com/es/neptune/

- SGBD potente, flexible y escalable y de propósito general
- Orientada a documentos, no relacional
  - Facilidad de uso
  - Escalable
  - Funcionalidad amplia
  - Optimizando la eficiencia (en tiempo)
- http://mongodb.org

- Existen versiones para múltiples plataformas: Linux, Windows, OS X
- Disponible como servicio: https://mms.mongodb.com
- Desarrollada en C++

- BD basada en documentos:
  - Los documentos (objetos) se acoplan perfectamente en los tipos de datos de lenguajes de programación
  - Los documentos incrustados y las colecciones reducen la necesidad de *joins*
  - Dispone de un esquema dinámico que facilita el polimorfismo
- Alto rendimiento:
  - Lecturas y escrituras rápidas
  - Índices potentes: sobre colecciones, subcolecciones y documentos incrustados
- Alta disponibilidad: replicación y restablecimiento automático del maestro
- Fácil escalabilidad:
  - *Sharding* automático
  - Lecturas eventualmente-consistentes

- Balanceo de carga: horizontalmente.
- Agregación
- Almacenamiento de archivos
- Ejecución de JavaScript en el servidor

Conceptos

- Documento: unidad básica de datos («fila»)
- Colección: grupo de documentos («tabla»)
- Una base de datos alberga múltiples colecciones
- Una instancia de MongoDB puede alojar múltiples bases de datos

Documento

Conjunto ordenado de pares <clave, valor>

```
{
 "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
 "Last Name": "PELLERIN",
 "First Name": "Franck",
 "Age": 29,
 "Address": {
    "Street": "1 chemin des Loges",
    "City": "VERSAILLES"
 }
}
```

Utilidades

- mongo: shell interactivo
- mongostat: Estadísticas de una instancia
- mongotop: tiempo de ejecución de operaciones
- monoimport/mongoexport
- mongodump/mongorestore

mongoshell

- Se comporta como shell de Unix
- Soporta expresiones en JavaScript
- Mandatos útiles:
  - help
  - db.help()
  - show dbs
  - use <database>
  - show collections
  - show users

MYSQL EXECUTABLE	ORACLE EXECUTABLE	MONGODB EXECUTABLE
mysqld	oracle	mongod
mysql	sqlplus	mongo

SQL TERM	MONGODB TERM
database	database
table	collection
index	index
row	document
column	field
joining	embedding & linking

SQL	MONGODB
CREATE TABLE users (name VARCHAR(128), age NUMBER)	db.createCollection("users")
INSERT INTO users VALUES ('Bob', 32)	db.users.insert({name: "Bob", age: 32})
SELECT * FROM users	db.users.find()
SELECT name, age FROM users	db.users.find({}, {name: 1, age: 1, _id:0})
SELECT name, age FROM users WHERE age = 33	db.users.find({age: 33}, {name: 1, age: 1, _id:0})
SELECT * FROM users WHERE age > 33	db.users.find({age: {$gt: 33}})
SELECT * FROM users WHERE age <= 33	db.users.find({age: {$lte: 33}})

SQL	MONGODB
SELECT * FROM users WHERE age > 33 AND age < 40	db.users.find({age: {$gt: 33, $lt: 40}})
SELECT * FROM users WHERE age = 32 AND name = 'Bob'	db.users.find({age: 32, name: "Bob"})
SELECT * FROM users WHERE age = 33 OR name = 'Bob'	db.users.find({$or:[{age:33}, {name: "Bob"}]})
SELECT * FROM users WHERE age = 33 ORDER BY name ASC	db.users.find({age: 33}).sort({name: 1})
SELECT * FROM users ORDER BY name DESC	db.users.find().sort({name: -1})
SELECT * FROM users WHERE name LIKE '%Joe%'	db.users.find({name: /Joe/})
SELECT * FROM users WHERE name LIKE 'Joe%'	db.users.find({name: /^Joe/})
SELECT * FROM users LIMIT 10 SKIP 20	db.users.find().skip(20).limit(10)
SELECT * FROM users LIMIT 1	db.users.findOne()

SQL	MONGODB
SELECT DISTINCT name FROM users	db.users.distinct("name")
SELECT COUNT(*) FROM users	db.users.count()
SELECT COUNT(*) FROM users WHERE AGE > 30	db.users.find({age: {$gt: 30}}).count()
SELECT COUNT(AGE) FROM users	db.users.find({age: {$exists: true}}).count()
UPDATE users SET age = 33 WHERE name = 'Bob'	db.users.update({name: "Bob"}, {$set: {age: 33}}, {multi: true})
UPDATE users SET age = age + 2 WHERE name = 'Bob'	db.users.update({name: "Bob"}, {$inc: {age: 2}}, {multi: true})
DELETE FROM users WHERE name = 'Bob'	db.users.remove({name: "Bob"})
CREATE INDEX ON users (name ASC)	db.users.ensureIndex({name: 1})
CREATE INDEX ON users (name ASC, age DESC)	db.users.ensureIndex({name: 1, age: -1})
EXPLAIN SELECT * FROM users WHERE age = 32	db.users.find({age: 32}).explain()

Ejemplos

```
x = 200

Math.sin(Math.PI / 2);

function factorial (n) {
  if (n <= 1) return 1;
  return n * factorial(n -1);
}

factorial(5)
```

```
db.runCommand( { create :
"jmbs", autoIndexId : true } )
```

Create

```
db.post = { "title" : "My Blog
Post",
     "content" : "Here's my
blog post.",
     "date" : new Date()}

db.blog.insert(post)

db.blog.find()
```

A co-Relational Model of Data for Large Shared Data Banks

- E. Meijer, G. Bierman
- ACM Queue, 2011

Contrary to popular belief, SQL and noSQL are really just two sides of the same coin

NoSQL Data Modeling Techniques

1. Conceptual Techniques
2. General Modeling Techniques
3. Hierarchy Modeling Techniques

Modeling Techniques

Cassandra

- Cassandra proporciona un sistema de almacenamiento escalable y con alta disponibilidad sin puntos de fallo
- Desarrollado por A. Y P. Malik para resolver su problema de búsqueda en la bandeja de entrada.
- Arquitectura basada en peer-to-peer frente a maestro-esclavo.
- “Tunable consistency”

Titan: Distributed Graph DB

- Titan is a scalable graph database optimized for storing and querying graphs containing hundreds of billions of vertices and edges distributed across a multi-machine cluster. Titan is a transactional database that can support thousands of concurrent users executing complex graph traversals in real time.

Titan (2)

- Elastic and linear scalability for a growing data and user base.
- Data distribution and replication for performance and fault tolerance.
- Multi-datacenter high availability and hot backups.
- Support for ACID and eventual consistency.

Titan (3)

- Support for various storage backends:
  - Apache Cassandra
  - Apache HBase
  - Oracle BerkeleyDB
- Support for global graph data analytics, reporting, and ETL through integration with big data platforms:

Machine Learning as a Service (MLaaS)

Machine Learning and Artificial Intelligence as a Service

- Servicios concretos para tareas de aprendizaje automático
- Gestión de datos
- Preprocesamiento
- Construcción de modelos
- Validación
- Despliegue

Propuestas en mercado

- OCCML
- AWS
- Azure
- Google
- Algorithmia

Algorithmia

- “Data sicentists never have to worry about infrastructure again”

Servicios de Google

- **AI Hub**: Componentes plug-and-play
- **Elementos básicos** (funcionalidad de visión, procesamiento de lenguaje natural, conversación, datos estructurados
- **AI Platform**: desarrollo basado en código:
  - Imágenes de MV con contenido (DL)
  - TPU
  - Kubeflow
  - Cuadernos
  - Escalado de modelos (serverless)

Storage as a Service

Almacenamiento como servicio

- Almacenamiento de archivos:
  - Caso de éxito: Dropbox
  - Opción Open-source: Owncloud (Nextcloud, Seafile)
- Almacenamiento de bloques

  IBM Cloud Block Storage

  NFS and SMB/CIFS access to cloud

Dropbox

- Inicios:
  - Pendrive en la nube
  - Alojados en AWS
- Migración a infraestructura propia: De Dropbox a sistema propio
- Crecimiento en servicios
  - Dropbox White paper

Owncloud

- Solución open-source para trabajo colaborativo, alternativa a Dropbox.
- Almacenamiento centralizado para documentos que se sincronizan con múltiples dispositivos
- Aplicación web y aplicaciones nativas
- Versiones: Community, online y Enterprise.

Couchdrop.io

- Couchdrop Cloud Gateway provides a dedicated on-prem or hosted, one click solution for accessing Cloud storage like Dropbox through NFS or the windows file sharing protocol SMB/CIFS.

Identity as a Service (IDaaS)

Identificación de usuarios

- La oferta de servicios a “todo” Internet implica ciertos desafíos en escalabilidad
- Gestión de identificación de usuarios y privilegios de acceso
- Elevado número de aplicaciones independientes usadas → explosión de credenciales → inseguridad

IDaaS

- Servicio (SaaS) de autenticación de usuarios y gestión de acceso a servicios
- Escalable
- Independiente de las aplicaciones funcionales posteriores
- Adaptable según necesidades de seguridad y coste (p.ej.: procesos biométricos)

Gestión de identidad y accesos en cloud

IAM en Cloud

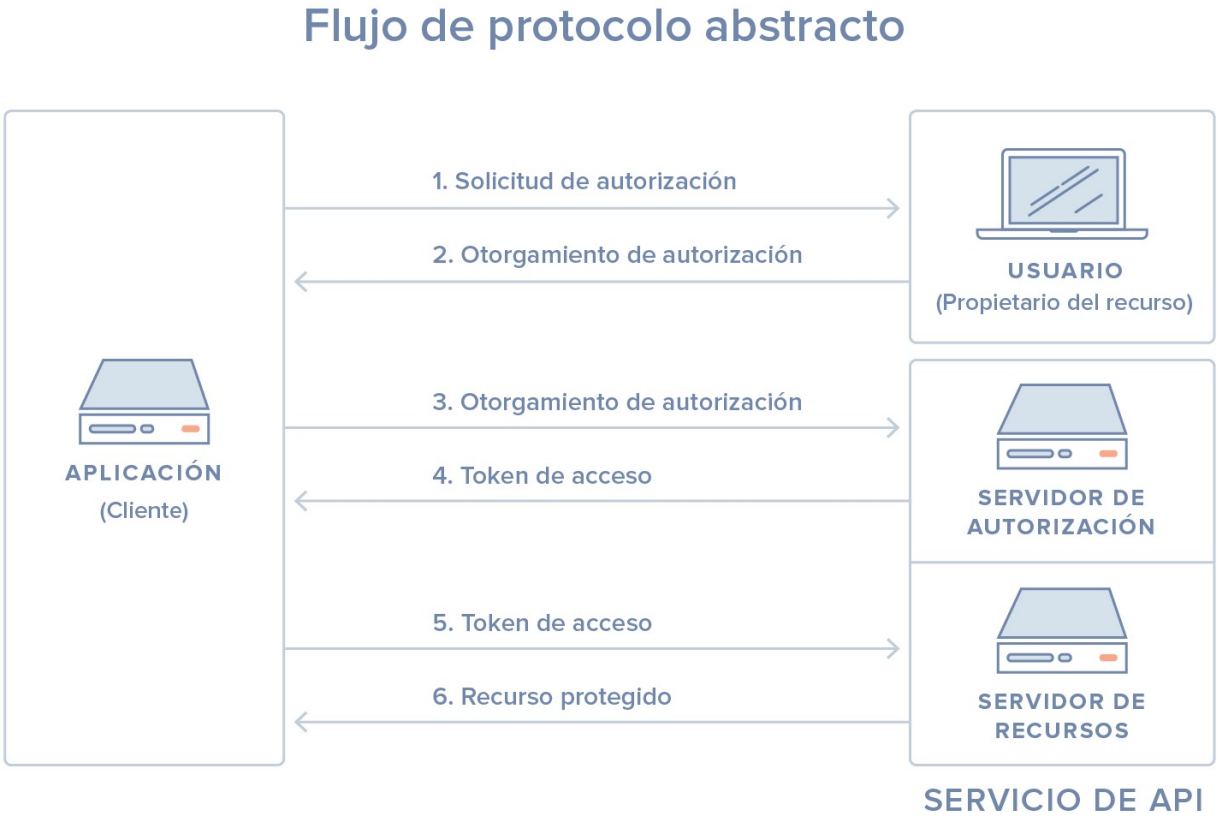
Lista de estándares en uso:

- **Security Assertion Markup Language** (SAML)
  Estándar para la administración de identidades federadas
- **OAuth**: Estándar para la autorización en servicios web
- **OpenID**: Estándar para autenticación federada en servicios web
- **eXtensible Access Control Markup Language** (XACML)
- **System for Cross-Domain Identity Management** (SCIM)

Conceptos

- Entidad
- Identidad
- Identificador
- Atributos
- Persona
- Papel
- Autenticación
- Control de acceso
- Autorización
- Fuente autoritativa
- Single Sign On

Flujo de protocolo OAuth



System for Cross-Domain Identity Management (SCIM)