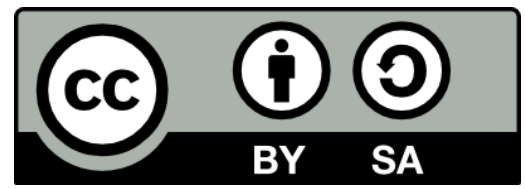
Big Data I:

Ingeniería de datos

Felipe Ortega
Dpto. de Estadística e Investigación Operativa
Universidad Rey Juan Carlos

March 23, 2015

Universidad
Rey Juan Carlos



Modelo de datos

- Define el diseño y la implementación del sistema que almacena y gestiona los datos.
- Datos estructurados.
  - Podemos almacenar sus valores en campos predefinidos con un tipo o una clase asociado de forma fija.
  - Ejemplo: sistemas de bases de datos relacionales (RDBMS).

Modelo de datos

- Datos no estructurados
- No podemos predefinir de antemano su tipo, por lo que necesitamos un modelo de datos más flexible para su gestión.
- Relaciones complejas entre los diferentes elementos de datos.
- Ejemplo: Sistemas NoSQL (particionado por columnas, documentos, grafos, clave-valor, etc.).

Bases de datos relacionales

- Numerosas opciones en el mercado, propietarias o software libre.
- Larga trayectoria, tecnología muy madura y consolidada, permite predecir hasta cierto punto rendimientos esperados.
  - Oracle, MySQL, MariaDB, PostgreSQL, SQLite, etc.
- Gran variabilidad en cuanto a soporte para big data.
  - Tipos de datos nativos.
  - Particionado de tablas.
  - Clustering, alta disponibilidad.
- Object-Relational Mapping (ORM).
  - Ejemplo: SQLAlchemy (Python).

Rendimiento RDBMS

- Es conveniente tener en cuenta algunas premisas importantes para analizar datos utilizando RDBMS.
- Utilizar motores *ACID-compliant* únicamente cuando sea imprescindible.
  - Si estamos trabajando con datos localmente, a los que solo accede uno o varios analistas, podemos usar otras opciones más rápidas.
  - Ejemplos: MyISAM (MySQL), ARIA (MariaDB).

- Desactivar claves primarias/foráneas en carga de datos. Después, utilizar solo cuando sea imprescindible.
- Activar particionado de tablas en diferentes ficheros (e.g. InnoDB).
- Cuidado con la codificación (asegurar UTF-8 siempre que sea posible).

Rendimiento RDBMS

- La configuración por defecto de cualquier servidor de base de datos relacional no suele ser adecuada para el análisis de datos.
  - Incrementar tamaño de los ficheros de claves (búsquedas, ordenación).
  - Configurar adecuadamente el sistema de logging (errores, consultas lentas, etc.) ya que puede consumir rápidamente espacio en disco.

- Formular las consultas basándonos siempre en conjuntos de datos y operaciones sobre los mismos (`JOIN`, `LEFT JOIN`, etc.).
  - Evitar en lo posible cláusulas del tipo `WHERE X IN (...)`, ya que suelen involucrar búcles de búsqueda lentos.
- Usar R y Python a partir de la fase de preparación y transformación.

NoSQL

- NoSQL = Not Only SQL.
- Escalabilidad y alto rendimiento para big data (en especial, tratamiento de tipos de datos muy heterogéneos o información textual).
- **Esquemas clave-valor**.
  - Almacen de datos en pares clave-valor, no precisan esquema (Riak, Redis, Voldemort, etc.).
- **Almacen de columnas**.
  - Particionan datos por columnas, de forma que podemos paralelizar consultas sobre subconjuntos de datos muy grandes (HBase, Cassandra).

Almacenamiento de datos: NoSQL

- **Orientadas a documentos**.
  - Cada clave está asociada a un documento, codificado según algún estándar de representación de datos (JSON, XML, YAML, etc.).
  - Los documentos pueden contener muchos pares clave-valor, clave-array (para listas de datos) u otros documentos.
  - Ejemplo: MongoDB.
- **Grafos**.
  - Almacenan explícitamente información sobre nodos y sus relaciones, optimizando consultas que recorren los grafos (Neo4J).
  - A cambio, tenemos que aprender un lenguaje de consultas muy diferente.

Ventajas NoSQL

- Mejor escalado horizontal, permite procesamiento paralelo.
  - Consideran la agregación de nuevos recursos de computación “en caliente” (autosharding, replicación).
- No es necesario definir esquemas (tipos de datos), se pueden mezclar dinámicamente nuevos datos con los ya existentes (con limitaciones).
- Mejor integración con metodologías ágiles de desarrollo de software.
  - Sprints cortos, prototipado rápido (vs. esquemas predefinidos).
  - Dificultad para establecer a priori esquemas fijos de estructuras de datos.
- Posibilidad de optimizar la recuperación de información mediante indexación (múltiple, dispersa, geoespacial, etc.).

Inconvenientes NoSQL

- Requieren importantes conocimentos técnicos para su instalación, correcta configuración y administración (recordemos importancia del rendimiento).
- Todavía escasa madurez en comparación con RDBMS.
- Múltiples estándares de programación y APIs, incompatibles entre sí en muchos casos (necesidad de soporte de comunicación).
- Problema sistemas distribuidos: teorema CAP (*consistencia*, *disponibilidad*, *particionado*).

Ejemplo: Redis

- Redis es un buen ejemplo de sistema de almacenamiento de datos NoSQL.
- Tremendamente popular, puesto que ofrece un buen rendimiento tanto con datos en memoria como con datos en disco.
- Buen enlace con Python: *redis-py*
  - `sudo pip install redis`
- Ejemplo de programación con Python y Redis (Python notebook).

Sistemas de ficheros distribuidos

- Adecuados para distribuir datos sobre clústers de máquinas/clouds.
- Fuertemente ligados a tecnologías específicas.
- HDFS (Apache Hadoop).
- Google File System (GFS).
- Amazon S3.