DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Bases de datos NoSQL

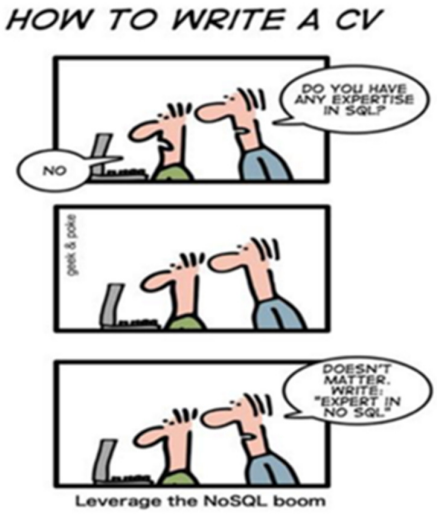
© Fernando Berzal, berzal@acm.org

Acceso a los datos

- Bases de datos relacionales: SQL
- O/R Mapping
- Bases de datos distribuidas
- Bases de datos NoSQL
- Bases de datos multidimensionales: Data Warehousing

NoSQL

- MapReduce
- Key-value stores
- Wide column stores
- Document stores
- Graph database systems



NoSQL

SQL = DBMS relacional (solución tradicional)
NoSQL = "**Not only SQL**" = DBMS no relacional

Motivación

Existen aplicaciones para las que las bases de datos relacionales no son la mejor solución…

… o para las cuales no todo se resuelve mejor usando exclusivamente una base de datos relacional.

NoSQL

Lo que ofrece un DBMS:

Característica	**DBMS relacional**
Conveniencia	Modelo de datos simple Lenguaje de consulta declarativo
Multi-usuario	Transacciones
Seguridad	Control de acceso a los datos
Fiabilidad	Replicación
Persistencia	Almacenamiento en ficheros
Volumen de datos ++	
Eficiencia (según para qué) +++	

NoSQL

Sistemas NoSQL

Alternativas a los DBMS relacionales

Pros:

- Flexibilidad a la hora de definir esquemas.
- Más sencillos de configurar.
- Más baratos.
- Escalabilidad.
- Consistencia relajada ➔ Mayor eficiencia/disponibilidad.

Contras:

- Sin lenguaje de consulta declarativo ➔ Más programación.
- Consistencia relajada ➔ Menores garantías.

NoSQL

Ejemplos de uso

Análisis de weblogs

- Registros (IP, timestamp, URL, …)
- Consultas altamente paralelizables.

Análisis de redes sociales

- Datos en forma de red (grafo con atributos).
- Consultas complejas (no adecuadas para SQL).

Wikipedia y otras colecciones de documentos

- Combinación de datos estructurados y no estructurados.
- Consultas y operaciones flexibles.

NoSQL

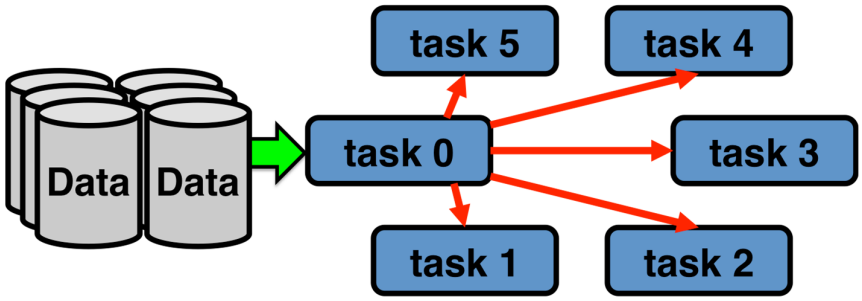
Alternativas de implementación

- Framework MapReduce ~ OLAP
- Key-value stores ~ OLTP
- Wide column stores
- Document stores
- Graph database systems

MapReduce

Aplicaciones

- "data-intensive": Grandes volúmenes de datos.
- "I/O-bound": más tiempo en acceder a los datos que en procesarlos (E/S domina sobre tiempo de CPU).

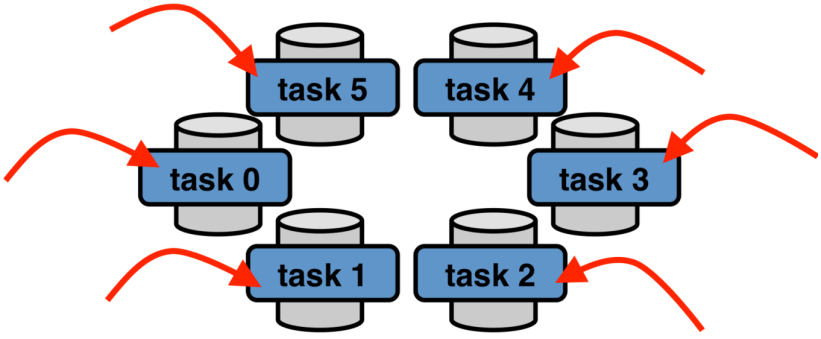


Solución tradicional (p.ej. MPI)
"bring the data to compute"

MapReduce

Aplicaciones

- "data-intensive": Grandes volúmenes de datos.
- "I/O-bound": más tiempo en acceder a los datos que en procesarlos (E/S domina sobre tiempo de CPU).



Solución MapReduce
"bring compute to the data"

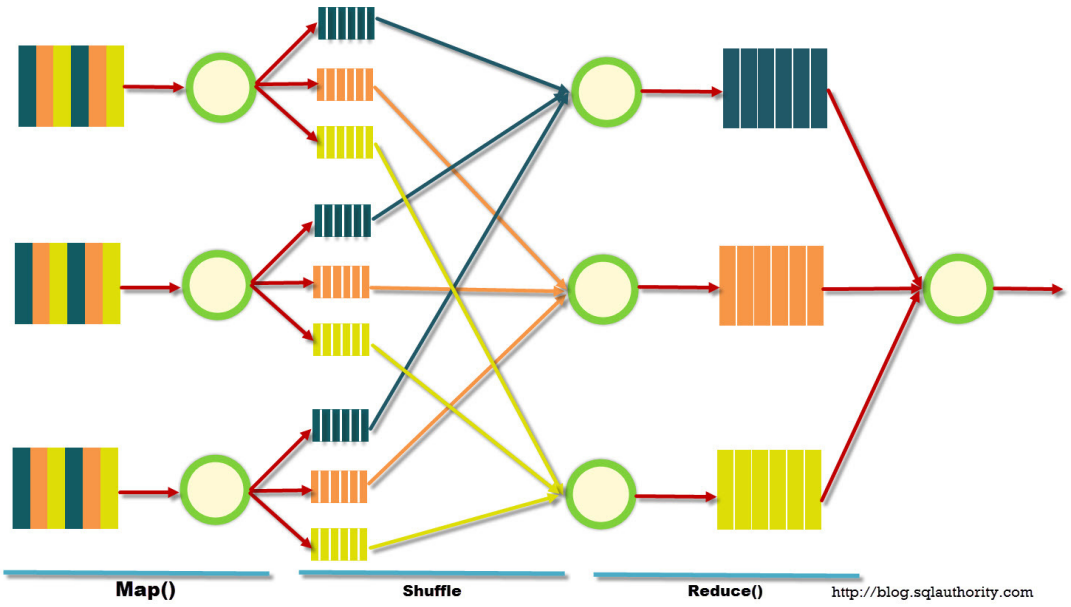
MapReduce

Modelo de programación cuya implementación permite procesar grandes cantidades de datos en un clúster utilizando algoritmos paralelos distribuidos.

Origen: Programación funcional
Funciones map y reduce de LISP (años 60): Vectorización

- map() permite realizar operaciones en paralelo sobre distintos fragmentos del conjunto de datos global.
- reduce() permite agregar/resumir los resultados obtenidos en paralelo.

MapReduce



MapReduce ≈ SELECT ... GROUP BY ...

MapReduce

Ejemplo

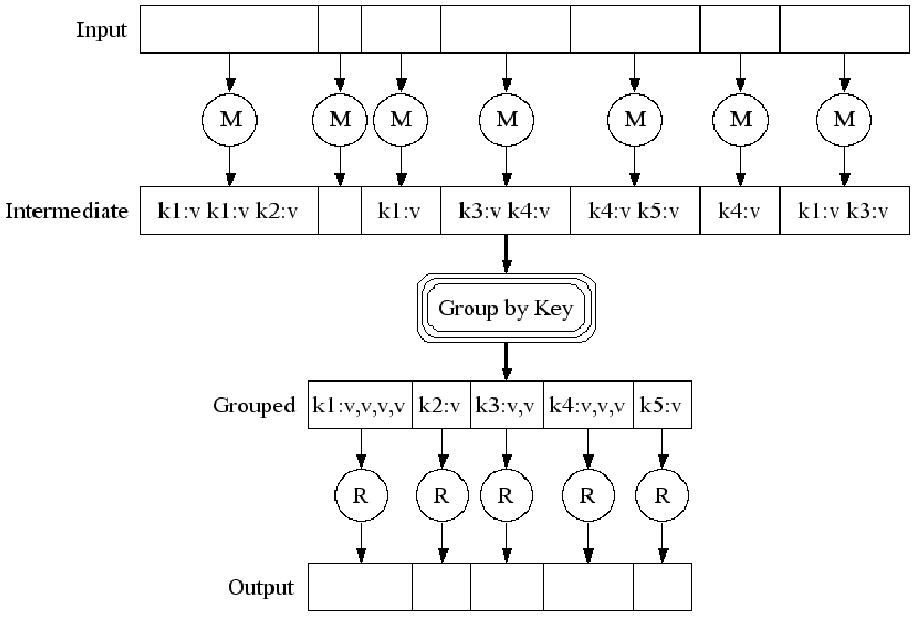
Contar la frecuencia de cada palabra en cada documento:

```
map(String input_key, String input_value):
    // input_key: document name
    // input_value: document contents
    for each word w in input_value:
        EmitIntermediate(w, "1");

reduce(String output_key, Iterator intermediate_values):
    // output_key: a word
    // output_values: a list of counts
    int result = 0;
    for each v in intermediate_values:
        result += ParseInt(v);
    Emit(AsString(result));
```


MapReduce

Ejemplo

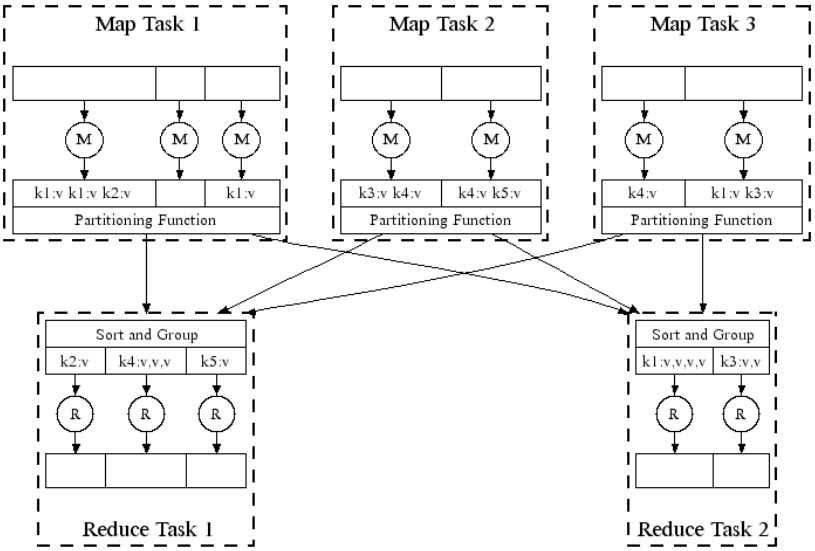
Ejecución



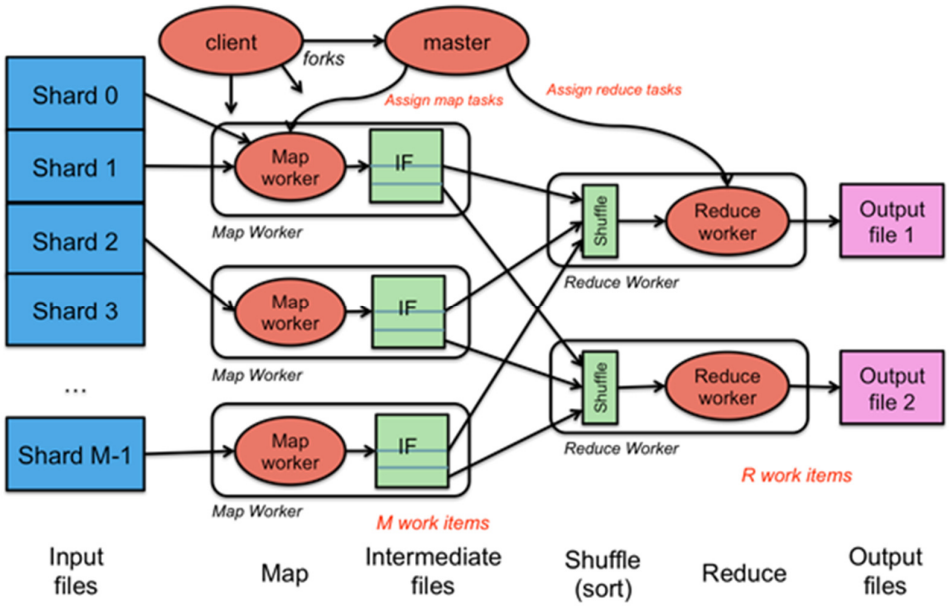
MapReduce

Ejemplo

Ejecución



MapReduce



MapReduce

Implementación

- Sin modelo de datos, sólo ficheros.
- Ofrece escalabilidad y tolerancia a fallos

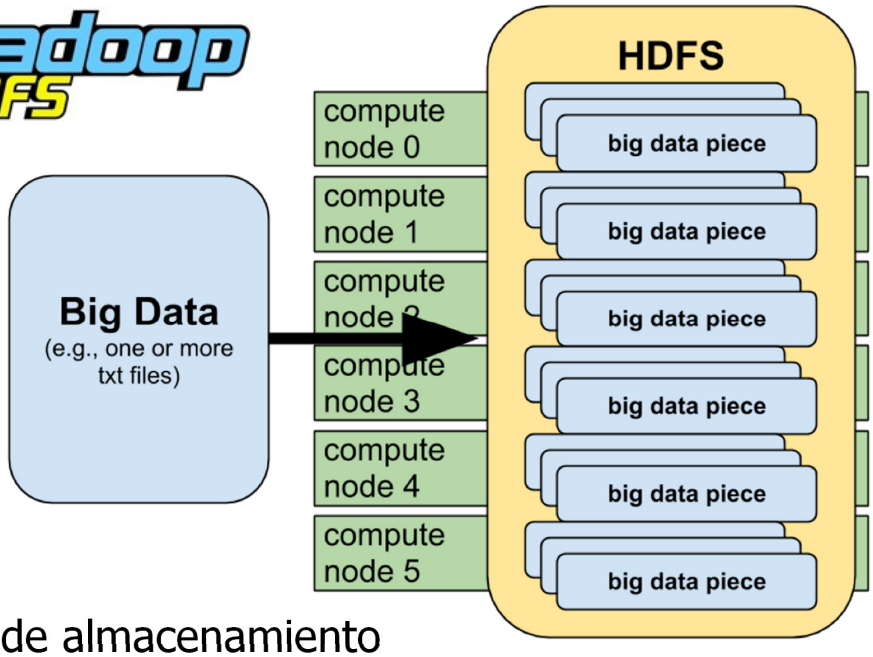
Implementación original: Google

- GFS [Google File System]

Implementación open-source: Hadoop (Yahoo!)

- HDFS [Hadoop Distributed File System]

MapReduce

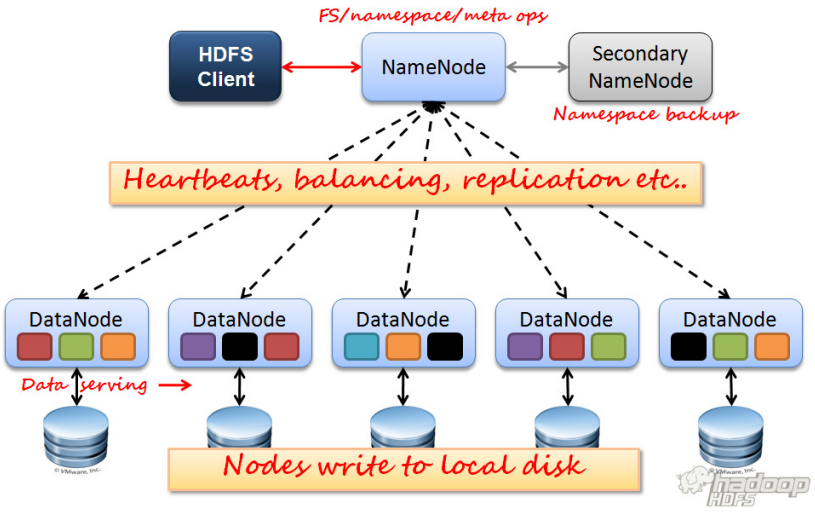


Sistema de almacenamiento distribuido, escalable y tolerante a fallos.

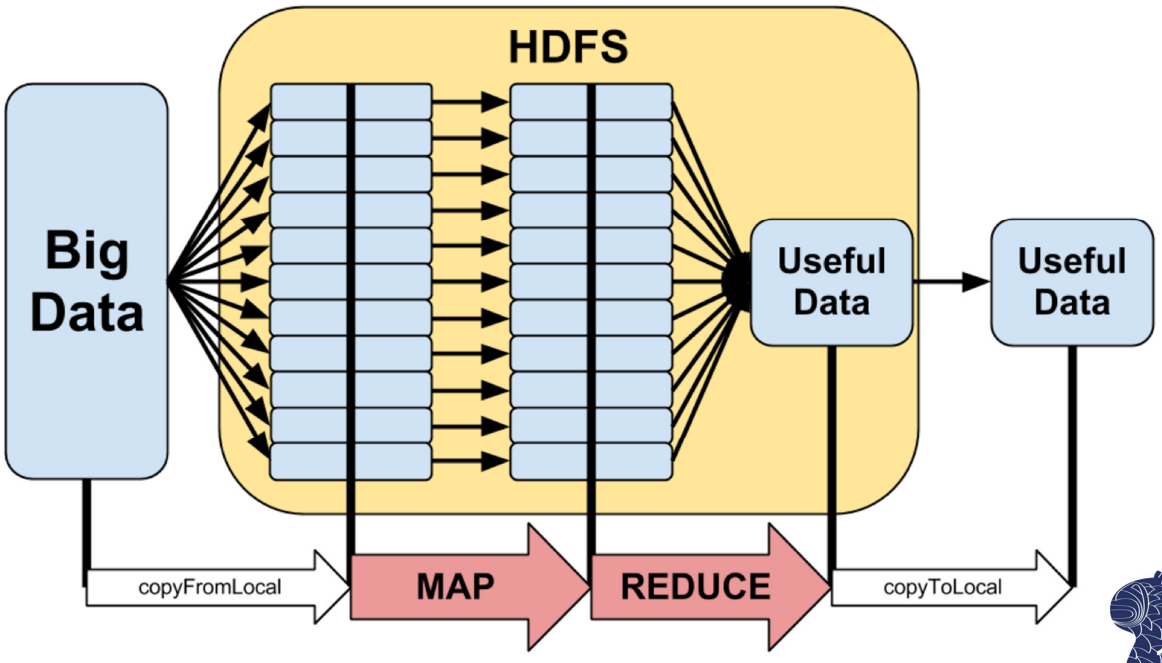
MapReduce

Tolerancia a fallos

- "heartbeat": ping periódico a cada tarea
- Replicación de datos:
  p.ej. 3 copias de cada fragmento de 64MB (Hadoop)



MapReduce



MapReduce

Almacenamiento de datos usando MapReduce:

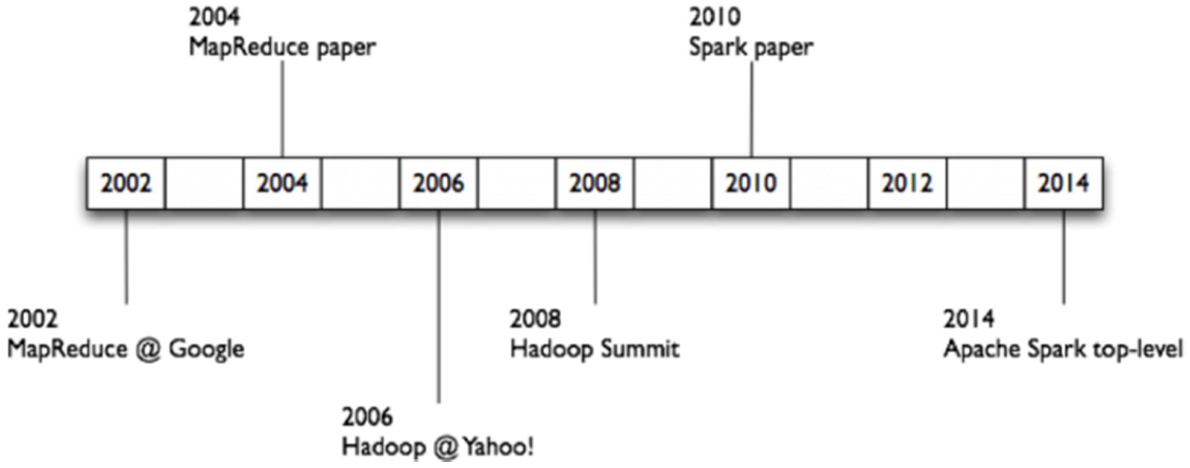
- **Apache Hive** (modelo relacional):
  Lenguaje de consulta HiveQL (similar a SQL)
  https://en.wikipedia.org/wiki/Apache_Hive

- **Apache Pig** (modelo más imperativo):
  Lenguaje Pig Latin
  https://en.wikipedia.org/wiki/Pig_(programming_tool)

- **Dryad** (Microsoft), similar a MapReduce/Hadoop:
  DryadLINQ (lenguaje de consulta integrado en un lenguaje de programación imperativo como C#).
  http://research.microsoft.com/en-us/projects/dryad/

MapReduce

Evolución de las soluciones MapReduce



2004

MapReduce: Simplified Data Processing on Large Clusters

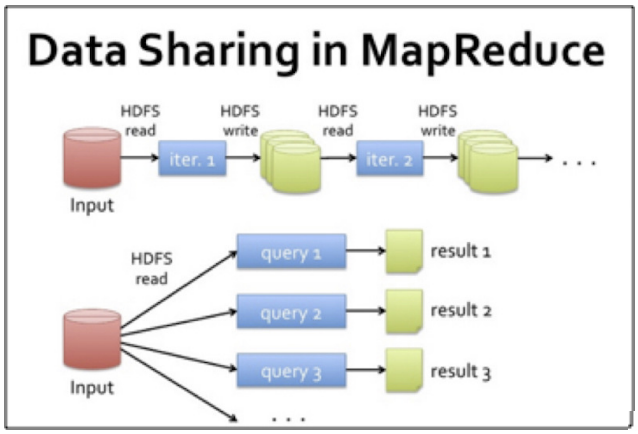
Jeffrey Dean and Sanjay Ghemawat

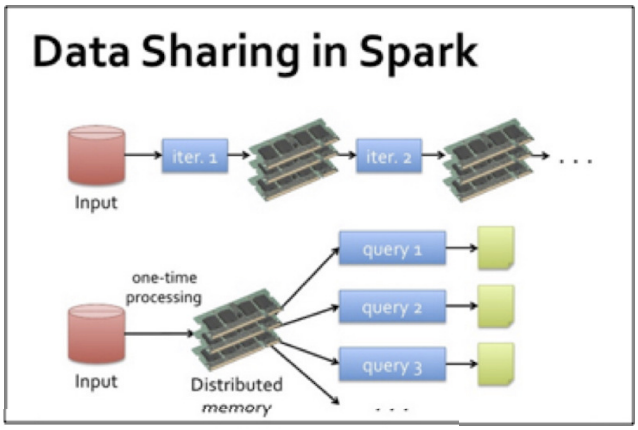
jeff@google.com, sanjay@google.com

Google, Inc.

MapReduce

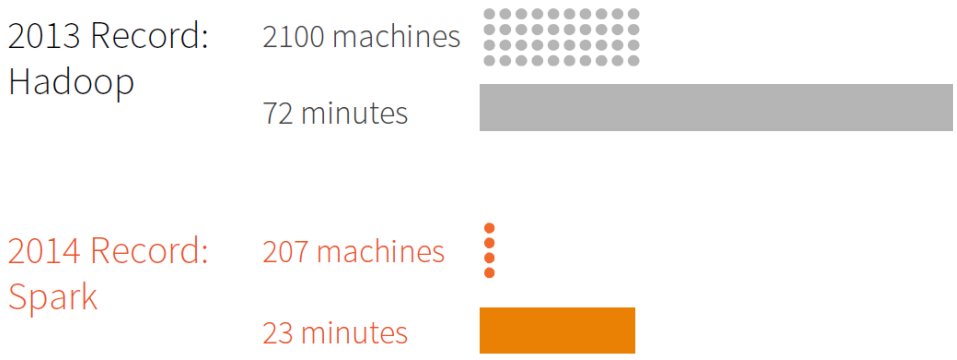
Hadoop vs. Spark





MapReduce

Benchmark
Ordenar 100TB en disco



Daytona Graysort Benchmark
http://sortbenchmark.org/

MapReduce

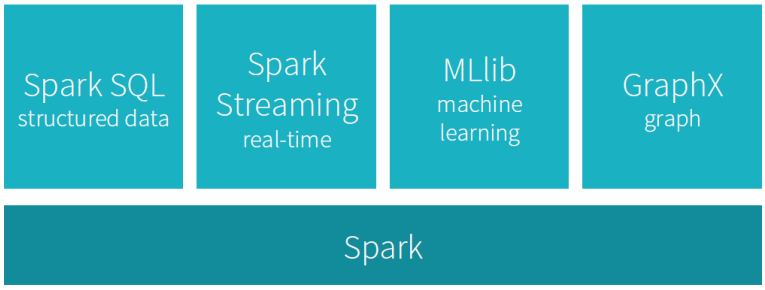
Apache Spark

http://spark.apache.org/

Framework unificado para distintos tipos de carga

MapReduce

Apache Spark

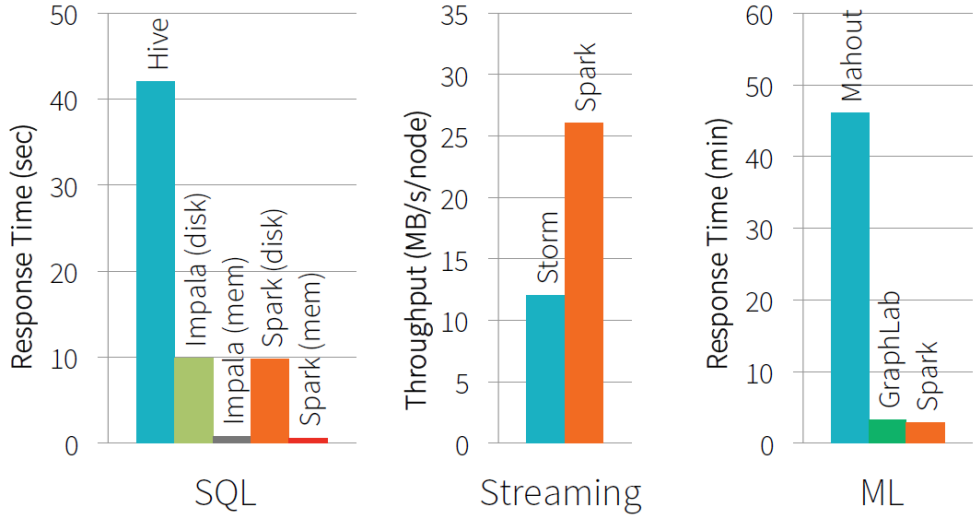


Spark:

Separate systems:

MapReduce

Apache Spark



Matei Zaharia:
"Making Big Data Processing Simple with Spark"
Databricks & MIT, December 2015

Key-value stores

OLTP [OnLine Transaction Processing]

DMBS con la interfaz más simple posible:

- Modelo de datos:
  pares <clave, valor>

- Operaciones: CRUD
  insert (key, value)
  fetch (key)
  update (key, value)
  delete (key)

Key-value stores

Implementación

Eficiente, escalable y tolerante a fallos

- Tabla hash distribuida: Registros almacenados en distintos nodos en función de su clave.
- Replicación de datos.
- Transacciones de un único registro: **"consistencia eventual"** (tras una actualización, eventualmente todos los accesos a la misma clave obtendrán el valor actualizado).

Key-value stores

Consistencia eventual: BASE vs. ACID

BASE [Basically Available, Soft state, Eventual consistency]

- Única garantía: "liveness".
- Incrementa la complejidad de los sistemas distribuidos.

ACID [Atomicity, Consistency, Isolation, Durability]

- Garantías tradicionales: "safety".
- DBMS relacional (commit/rollback).

Key-value stores

Ejemplos

- **Redis** (ANSI C)
  http://redis.io/

- **Memcached**
  http://memcached.org/

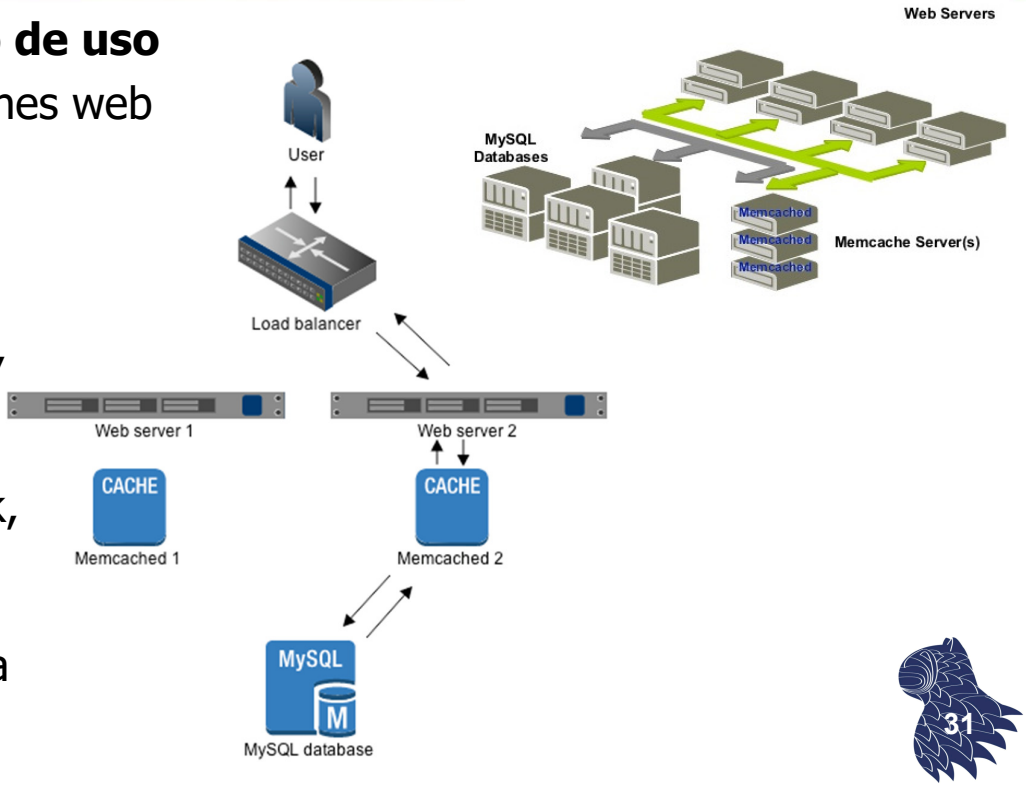
- **Amazon DynamoDB**
  http://aws.amazon.com/dynamodb/

Key-value stores

Ejemplo de uso

Aplicaciones web

p.ej.
YouTube,
Reddit,
Zinga,
Facebook,
Twitter,
Tumblr,
Wikipedia
...



Key-value stores

Algunas implementaciones permiten ordenar las claves, lo que permite realizar consultas sobre rangos de valores y procesar las claves en orden.

Muchos sistemas de este tipo incluyen extensiones que los acercan a otros tipos de sistemas NoSQL:

- Wide column stores
- Document stores

Wide column stores

Como en las bases de datos relacionales, los datos se almacenan en tablas, con filas y columnas.

A diferencia de las bases de datos relacionales, los nombres y el formato de las columnas puede variar de una fila a otra dentro de la misma tabla

Row ID	Columns...		
1	Name	Website	
	Preston	www.example.com	
2	Name	Website	
	Julia	www.example.com	
3	Name	Email	Website
	Alice	example@example.com	www.example.com

Wide column stores

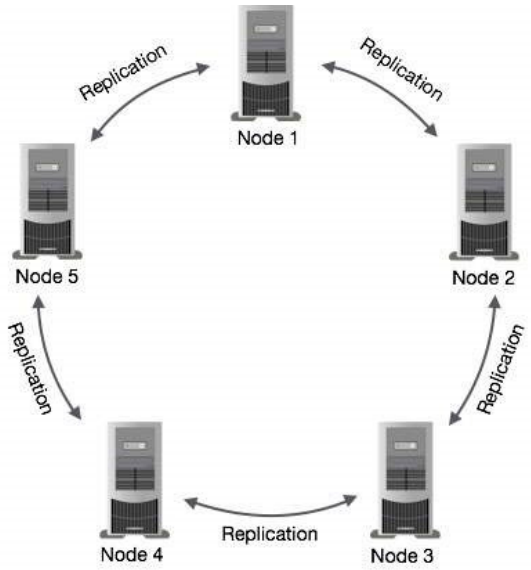
Ejemplos

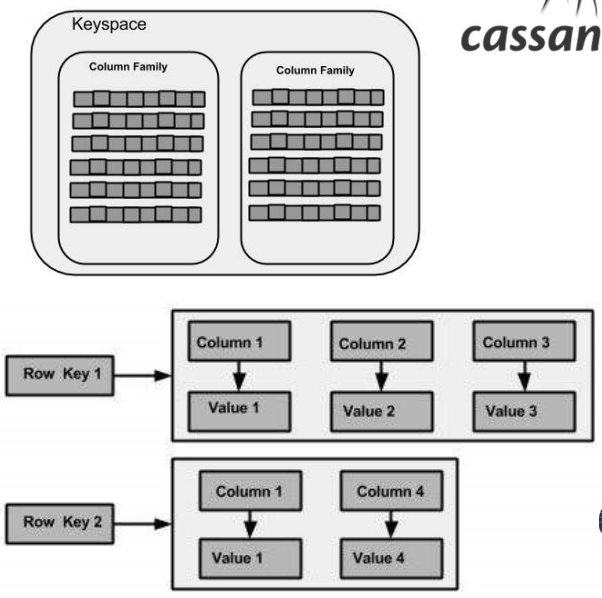
- **Google BigTable** (OSDI'2006)
  https://cloud.google.com/bigtable/
- **Apache Cassandra** (Facebook)
  http://cassandra.apache.org/
- **Apache HBase** (Java, sobre HDFS, como Google BigTable sobre GFS)
  http://hbase.apache.org/
- **Apache Accumulo** (NSA)
  https://accumulo.apache.org/

Wide column stores

Apache Cassandra

Architectura P2P





Wide column stores

Apache Cassandra

CQL [Cassandra Query Language]

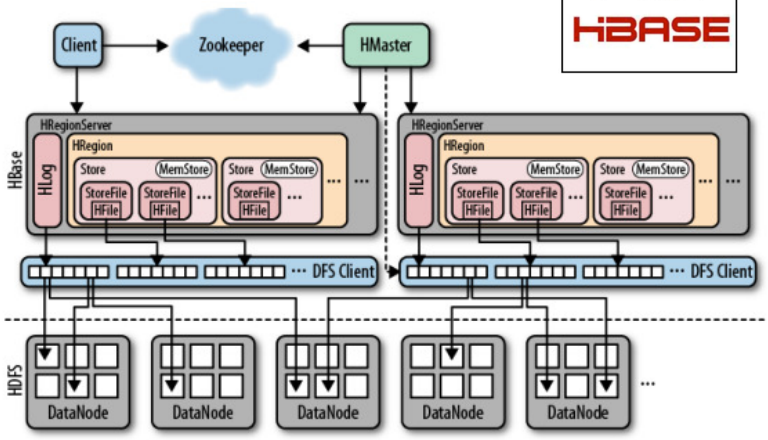
http://www.tutorialspoint.com/cassandra/

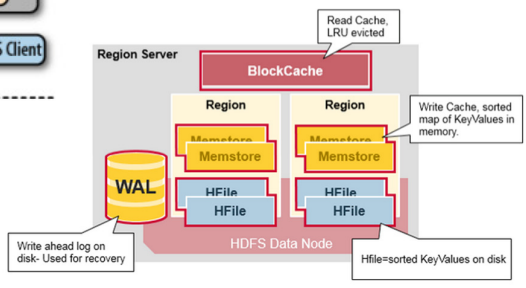
```
CREATE KEYSPACE MyKeySpace
  WITH REPLICATION = { 'class' : 'SimpleStrategy',
                       'replication_factor' : 3 };
USE MyKeySpace;

CREATE COLUMNFAMILY MyColumns
   (id text, Last text, First text, PRIMARY KEY(id));

INSERT INTO MyColumns (id, Last, First)
               VALUES ('1', 'Doe', 'John');

SELECT * FROM MyColumns;
```

Usuarios: Apple (>10PB, 100 000 nodos), Netflix…

Wide column stores

Apache HBase





Información detallada…

https://www.mapr.com/blog/in-depth-look-hbase-architecture

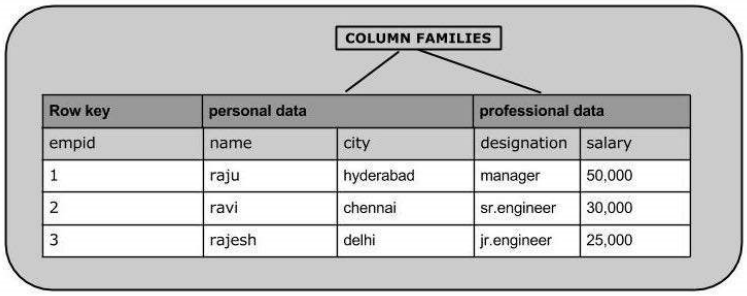
Wide column stores

Apache HBase

HBase shell

http://www.tutorialspoint.com/hbase/

```
create 'emp', 'personal data', 'professional data'
put 'emp','1','personal data:name','Jose'
put 'emp','1','personal data:city','Granada'
put 'emp','1','professional data:designation','manager'
…
scan 'emp'
```



Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

Usuarios: Facebook (desde 2010), LinkedIn, Spotify…

Document stores

a.k.a. document-oriented databases

No existe un esquema de la base de datos:

- Cada registro puede tener una estructura diferente.
- Tipos de las columnas variables de un registro a otro.
- Las columnas pueden tener más de un valor (arrays).
- Los registros pueden tener estructura propia [nested].

Representación de los datos utilizando JSON o XML.

Document stores

Implementación

- Como los almacenes clave-valor, salvo que ahora el valor es un documento semiestructurado (JSON, XML).
- Operaciones básicas de un almacén clave-valor:
  - insert (key, value)
  - fetch (key)
  - update (key, value)
  - delete (key)
- Consultas limitadas sobre el contenido de los documentos (dependientes del sistema concreto).

Document stores

Ejemplos más populares

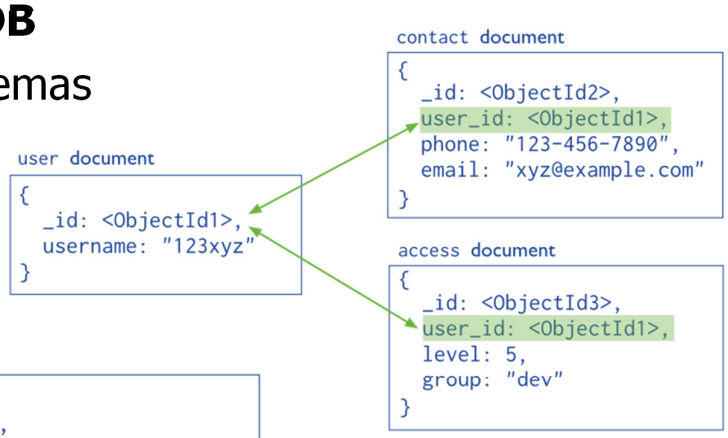
- **MongoDB** (C/C++, Javascript)
  https://www.mongodb.org/
- **Couchbase** (C/C++, Erlang)
  http://www.couchbase.com/
- **CouchDB** (Erlang)
  http://couchdb.apache.org/
- **Google Datastore**
  https://cloud.google.com/datastore/
- **Amazon DynamoDB**
  http://aws.amazon.com/dynamodb/
- **MarkLogic**
  http://www.marklogic.com/

Document stores

MongoDB

Sin esquemas

JSON




```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
            level: 5,
            group: "dev"
          }
}
```

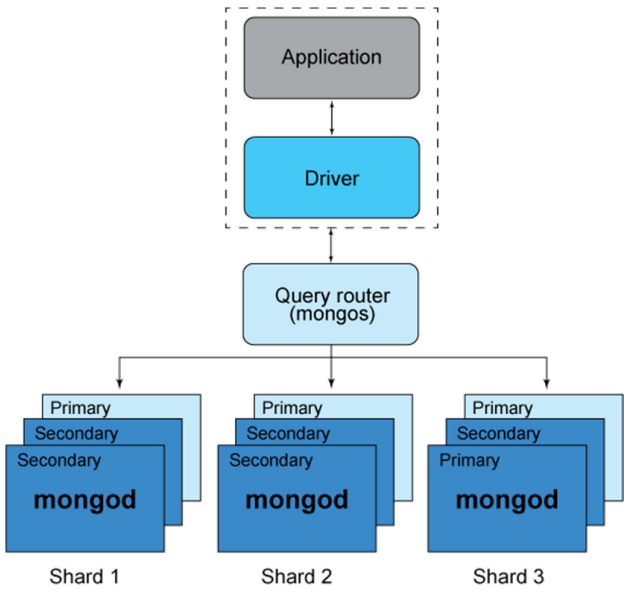
Embedded sub-document

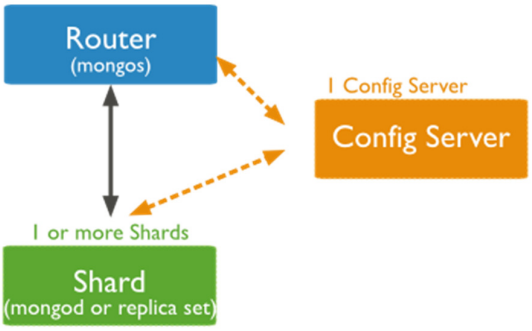
Embedded sub-document

Document stores

MongoDB

Arquitectura

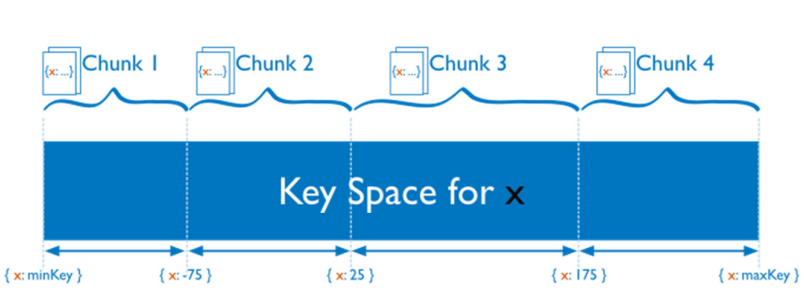


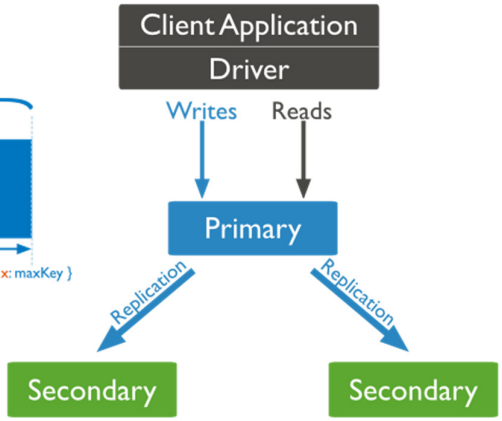


Document stores

MongoDB

Sharding

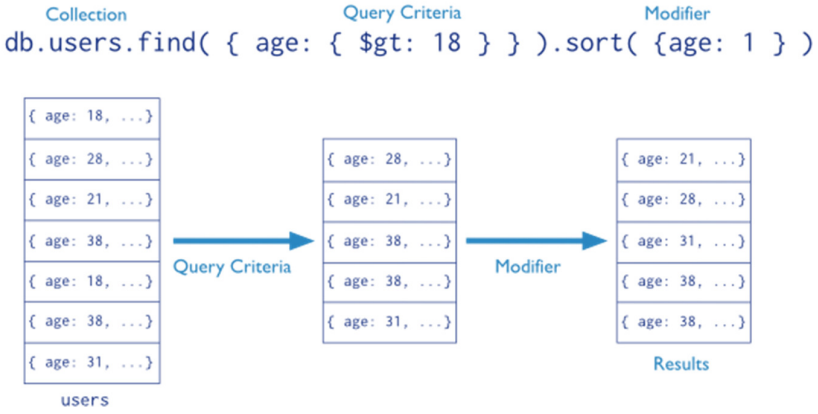




Document stores

MongoDB

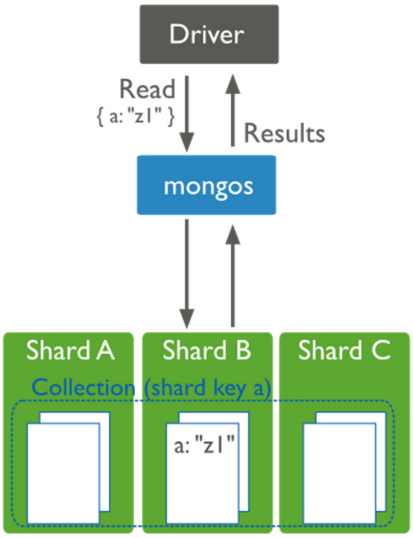
CRUD: Consultas

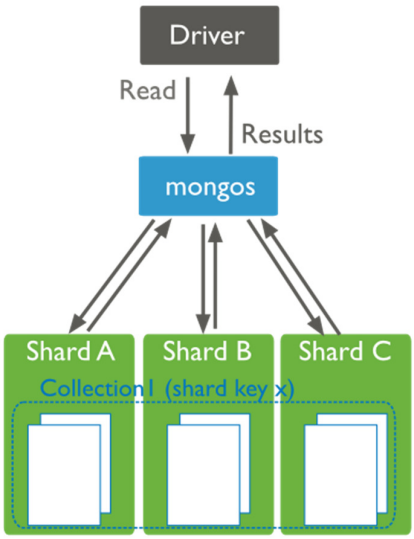


Document stores

MongoDB

CRUD: Consultas

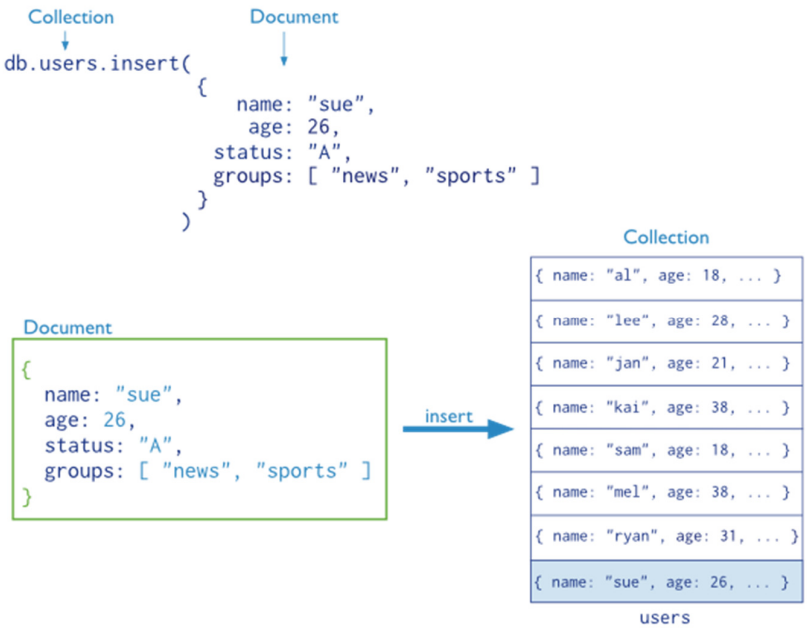




Document stores

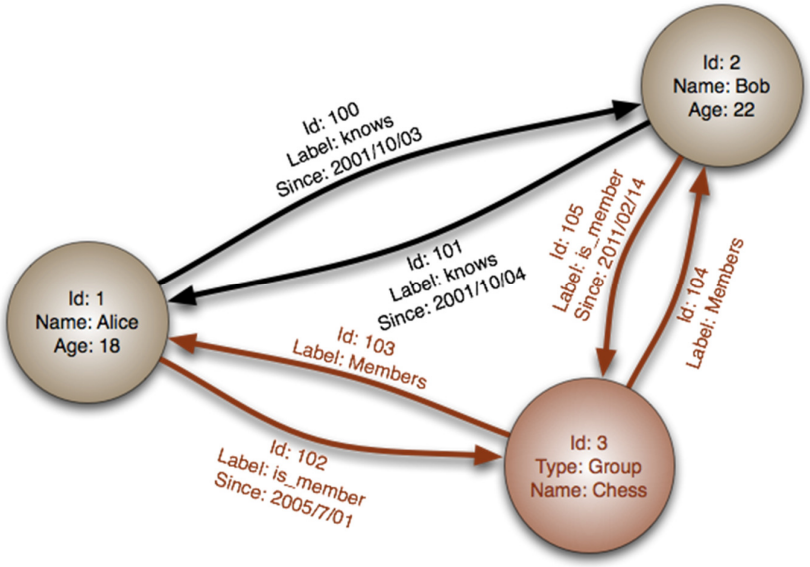
MongoDB

CRUD: Actualizaciones (insert|update|remove)



Graph database systems

Utilizan grafos con atributos para almacenar los datos:



Graph database systems

Sistemas

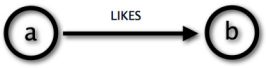
- **Neo4j** (Java)
  http://neo4j.com/
- **OrientDB** (Java, multi-modelo)
  http://orientdb.com/
- **Titan** (Java)
  http://thinkaurelius.github.io/titan/

Graph database systems

Neo4j

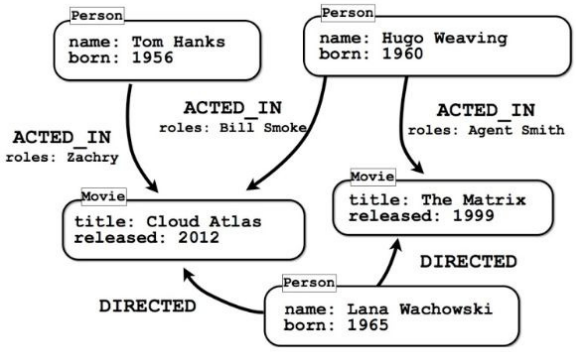
Cypher (query language)

http://neo4j.com/docs/stable/cypher-refcard/



Cypher

```
(a) -[:LIKES]-> (b)
```

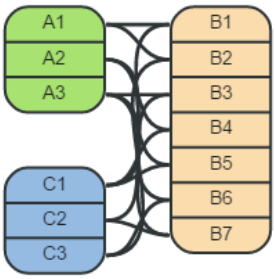



```
MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)
WHERE movie.title =~ "T.*"
RETURN movie.title as title,
       collect(actor.name) as cast
ORDER BY title ASC LIMIT 10;
```

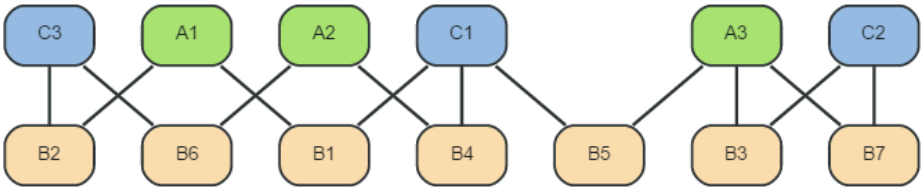

Graph database systems

neo4j

Relational DB



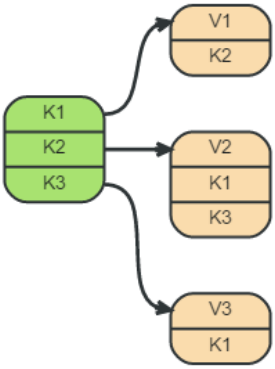
Graph DB



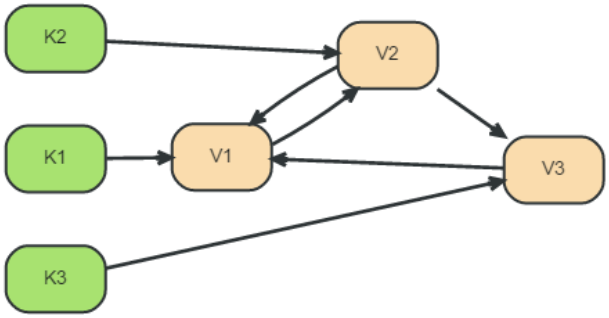
Graph database systems

neo4j

Key-value store



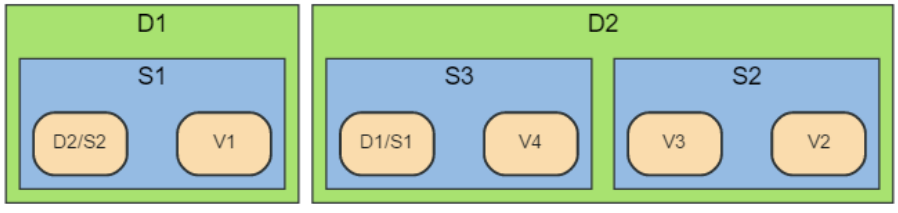
Graph DB



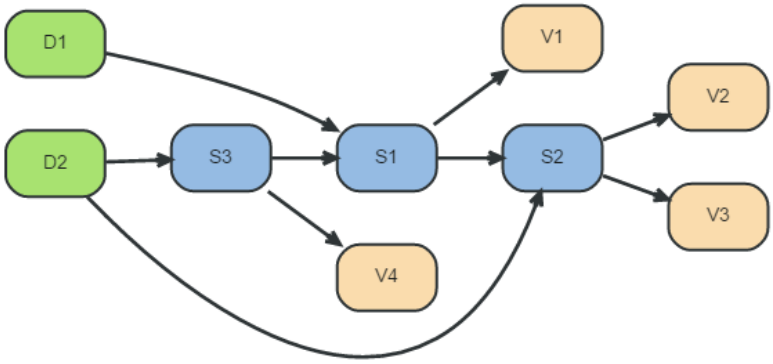
Graph database systems

neo4j

Document store



Graph DB

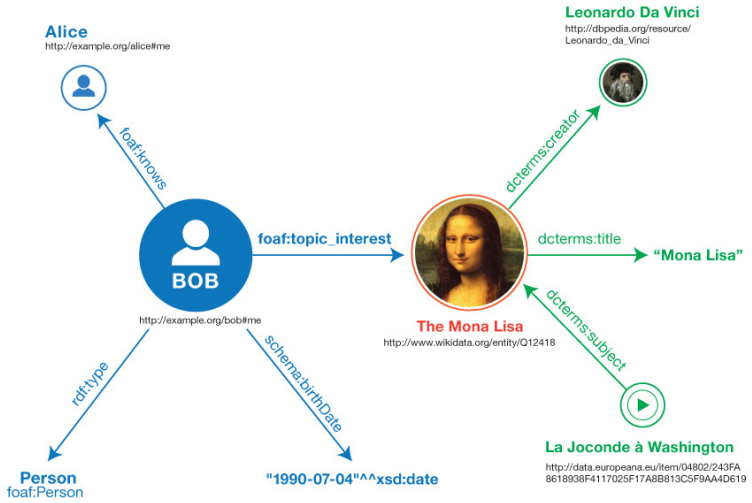


Graph database systems

Ejemplo: Web semántica

Tripletas RDF [Resource Description Framework]

`<subject> <predicate> <object>`



Graph database systems

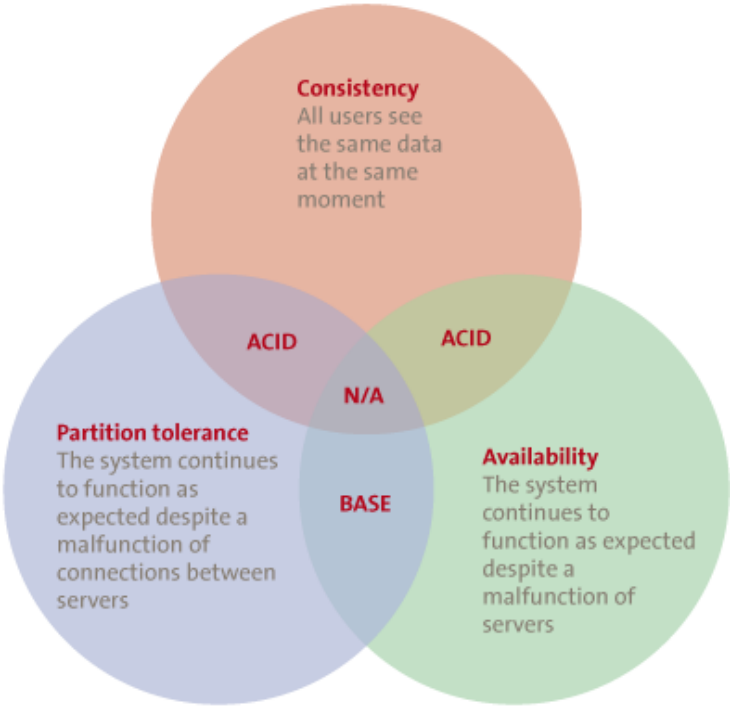
Ejemplo: Web semántica

Lenguaje de consulta SPARQL ["sparkle"]
p.ej. Neo4j, Virtuoso…

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

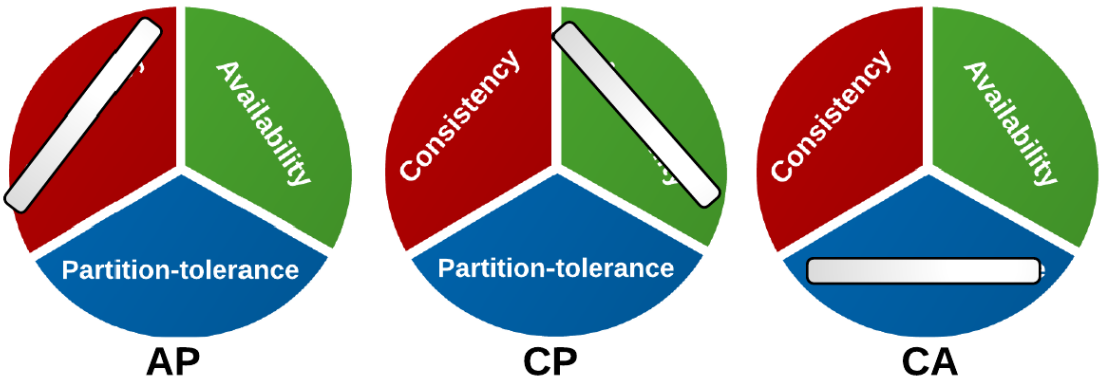
Acrónimo recursivo:
SPARQL Protocol and RDF Query Language

Bases de datos NoSQL



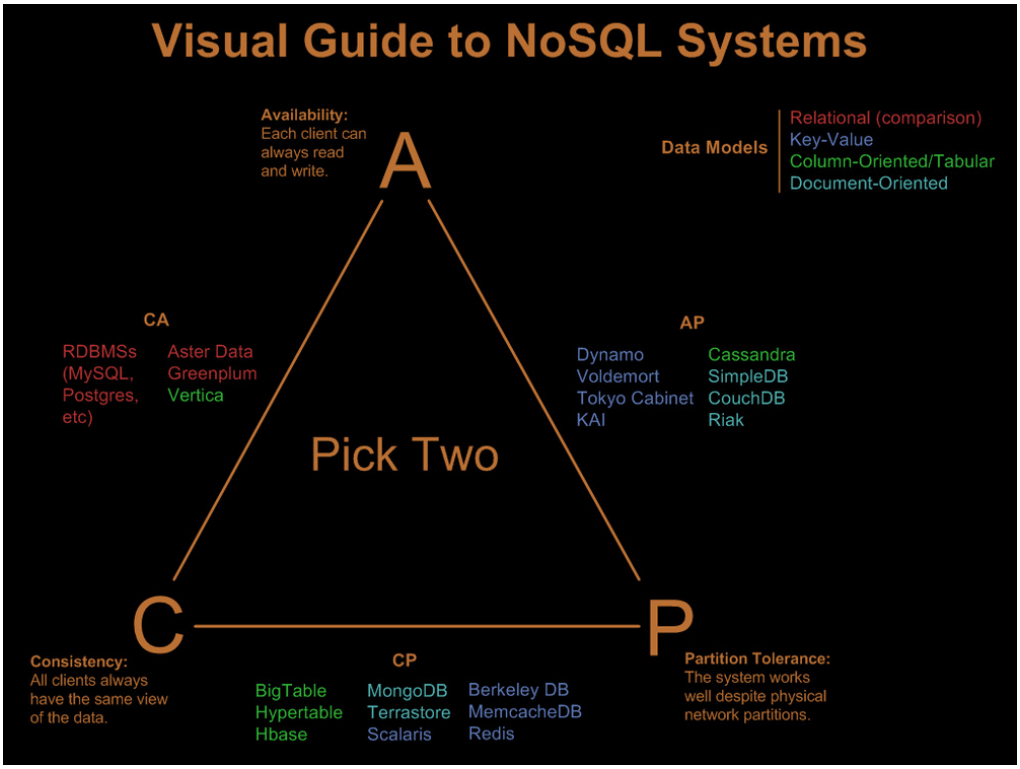
Bases de datos NoSQL

Teorema CAP



http://captheorem-jweaver.rhcloud.com/

Bases de datos NoSQL



Bases de datos NoSQL

MongoDB

- C++
- Distribución: distintos nodos con réplicas de los datos.
- **Consistencia estricta**: uno de los nodos ejerce de nodo primario (todas las operaciones de escritura), los demás son nodos secundarios.

CouchDB

- Erlang
- Distribución: distintos nodos con réplicas de los datos.
- **Consistencia eventual**: se permiten operaciones de escritura sin esperar la confirmación de los demás nodos (copias incrementales de los cambios).

Bases de datos NoSQL

Compromiso consistencia-disponibilidad

Más importante garantizar la consistencia de los datos…

mongoDB

APACHE HBASE

Más importante la disponibilidad de los datos…

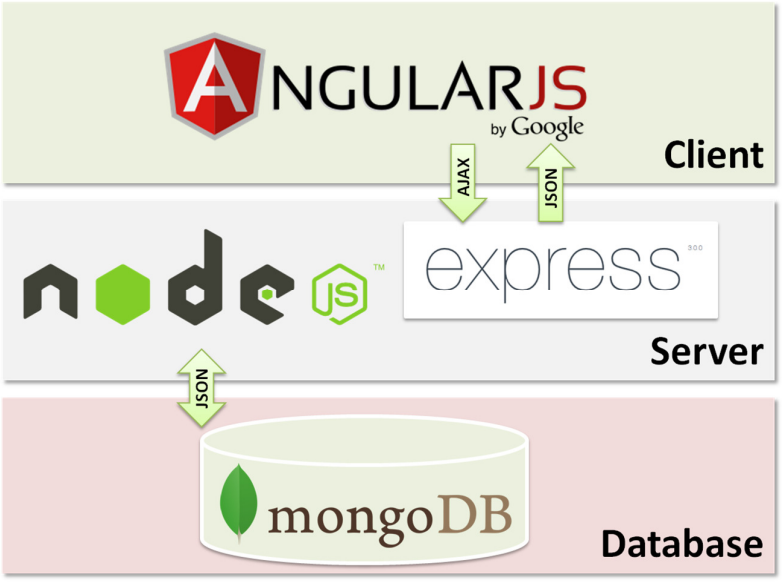
CouchDB relax

cassandra

Arquitecturas típicas

MEAN stack

http://mean.io/#!/



mongoDB

express

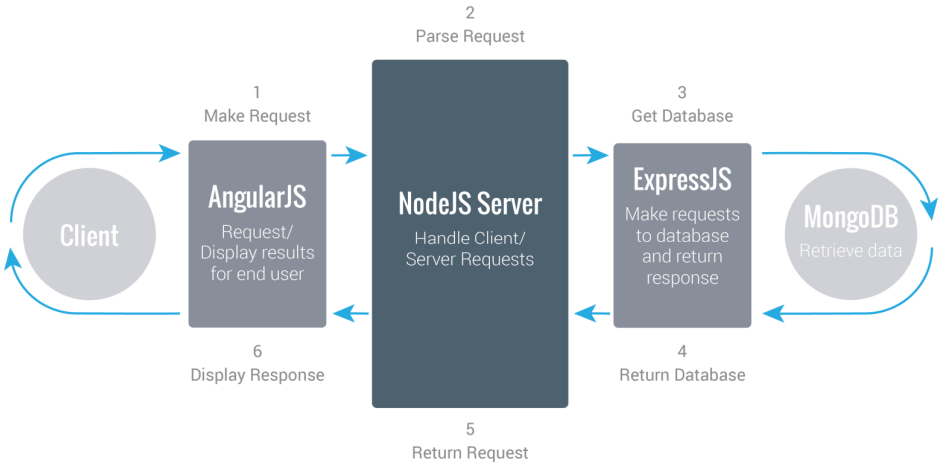
ANGULARJS by Google

node

Arquitecturas típicas

MEAN stack

http://mean.io/#!/



Arquitecturas típicas

LYME/LYCE stack

- Linux
- Yaws (web server)
- Mnesia/CouchDB (database)
- Erlang (programming language)