PLASMAC

Plataforma en la nube para la mejora de la atención socioasistencial de la Macaronesia

Actividad 2.2.2: Análisis y diseño de los procesos de liberación de datos e información

Arquitectura de Sistemas de Datos

[Escriba aquí una descripción breve del documento. Normalmente, una descripción breve es un resumen corto del contenido del documento. Escriba aquí una descripción breve del documento. Normalmente, una descripción breve es un resumen corto del contenido del documento.]

Descargo de Responsabilidad

Este documento ha sido preparado por el personal de Instituto Tecnológico y de Energías Renovables S. A. (ITER). ITER es una empresa pública con amplia experiencia en el desarrollo de proyectos relacionados con el uso de las TIC.

Este informe representa el mejor criterio de ITER a la luz de la información disponible. Se informa al lector que, dado el amplio alcance del proyecto, no toda la información se puede verificar o comprobar de forma independiente y que, por lo tanto, algunas afirmaciones en el informe pueden basarse en una única fuente de información. Aunque se usa el tiempo condicional para resaltar el grado de incertidumbre, el lector debe entender que el uso de la información contenida en este informe es responsabilidad del lector.

Glosario de términos

ACID (Atomicity, Consistency, Isolation and Durability)	Atomicidad, Consistencia, Aislamiento y Durabilidad
API (Application Programming Interface)	Interfaz de Programación de Aplicaciones
BI (Business Intelligence)	Inteligencia de Negocio
DAG (Directed Acyclic Graph)	Grafo Acíclico Dirigido
DBMS (DataBase Managemente System)	Sistema Gestor de Base de Datos
DW (Data Warehouse)	Almacén de Datos
ETL (Extract, Transform and Load)	Extraer, Transformar y Cargar
HDFS (Hadoop Distributed File System)	Sistema de Archivos Distribuido de Hadoop
JSON (JavaScript Object Notation)	Notación de Objeto de JavaScript
OLAP (OnLine Analytical Processing)	Procesamiento Analítico en Línea
OLTP (OnLine Transaction Processing)	Procesamiento de Transacciones en Línea
RDBMS (Relational DataBase Managemente System)	Sistema Gestor de Base de Datos Relacional
RDD (Resilient Distributed DataSet)	Conjunto de datos distribuido y resiliente.
SQL (Structured Query Language)	Lenguaje de Consulta Estructurada
YARN (Yet Another Resource Negotiator)	Otro Negociador de Recursos

Tabla de contenido

1 Arquitectura de un sistema de Big Data

La arquitectura de un sistema de Big Data está diseñada para manejar la ingestión, el almacenamiento, el procesamiento y el análisis de los datos que, por volumen, velocidad o variedad, no puede realizarse mediante sistemas tradicionales. Generalmente involucran uno o más de los siguientes tipos de carga de trabajo:

- Ingesta y almacenamiento.
- Procesamiento por lotes.
- Procesamiento en tiempo real.
- Aprendizaje automático.
- Análisis y generación de informes.

A continuación se muestran los componentes de la arquitectura de un sistema de Big Data.

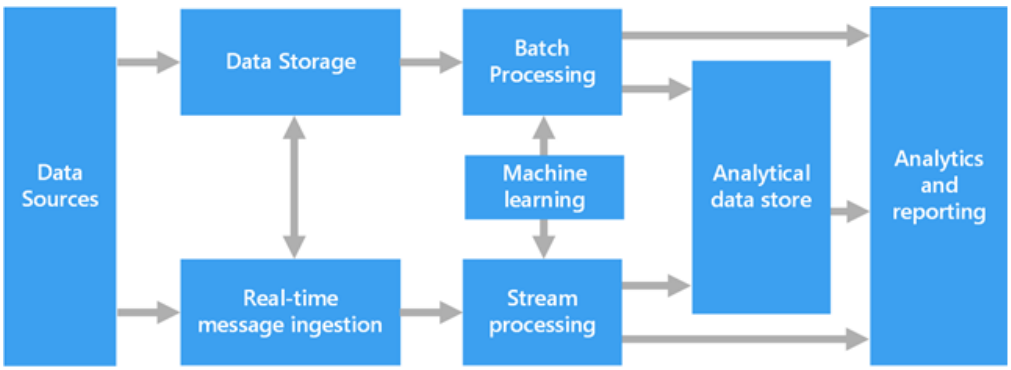

Figura 1. Arquitectura de un sistema de Big Data.

Dependiendo de la solución, la arquitectura incluye algunos o todos de los siguientes componentes:

- Fuentes de datos: almacenamiento de la información de la que extraer conocimiento. Entre las más populares destacan los sistemas de base de datos relaciones y no relacionales y las fuentes de datos en tiempo real como los dispositivos IoT.
- Almacenamiento: almacena y replica grandes volúmenes de datos mediante tecnologías distribuidas de almacenamiento.
- Procesamiento por lotes: transforma grandes volúmenes de datos mediante trabajos por lotes de larga ejecución que requieran un acceso al conjunto completo de los datos.
- Ingesta en tiempo real: captura y almacena los datos en tiempo real actuando como buffer de flujo para los distintos mensajes, permitiendo así el escalado horizontal de su procesamiento o su entrega confiable.
- Procesamiento en tiempo real: transforma los datos en tiempo real de forma apropiada para su posterior análisis.

- Aprendizaje automático: desarrolla algoritmos capaces de generalizar comportamientos a partir de los datos procesados.
- Almacén analítico de datos: almacenamiento de los datos procesados en un formato estructurado, permitiendo así su consulta a través de herramientas analíticas. Puede ser de tipo relacional como en la mayoría de soluciones tradicionales de inteligencia de negocio o alternativamente una tecnología no relacional (NoSQL) de datos distribuidos de baja latencia.
- Análisis y generación de informes: extrae conocimiento a través de la exploración interactiva y visual de los datos procesados.

1.1 Sistemas de Base de Datos

Una base de datos es una colección lógica de información que se administra mediante un software específico o sistema de gestión de base de datos (DBMS). Ambos forman un sistema de base de datos que incluye no solo la información de usuario sino también los objetos necesarios para su administración como, por ejemplo, índices o archivos de registro.

Los DBMS que siguen un modelo de datos relacional (entidad-relación) se denotan como RDBMS. En caso de que sean compatibles con otros modelos de datos se denotan como NoSQL.

A continuación se muestra el comportamiento en el tiempo de la popularidad de distintos DBMS, tanto RDBMS como NoSQL.

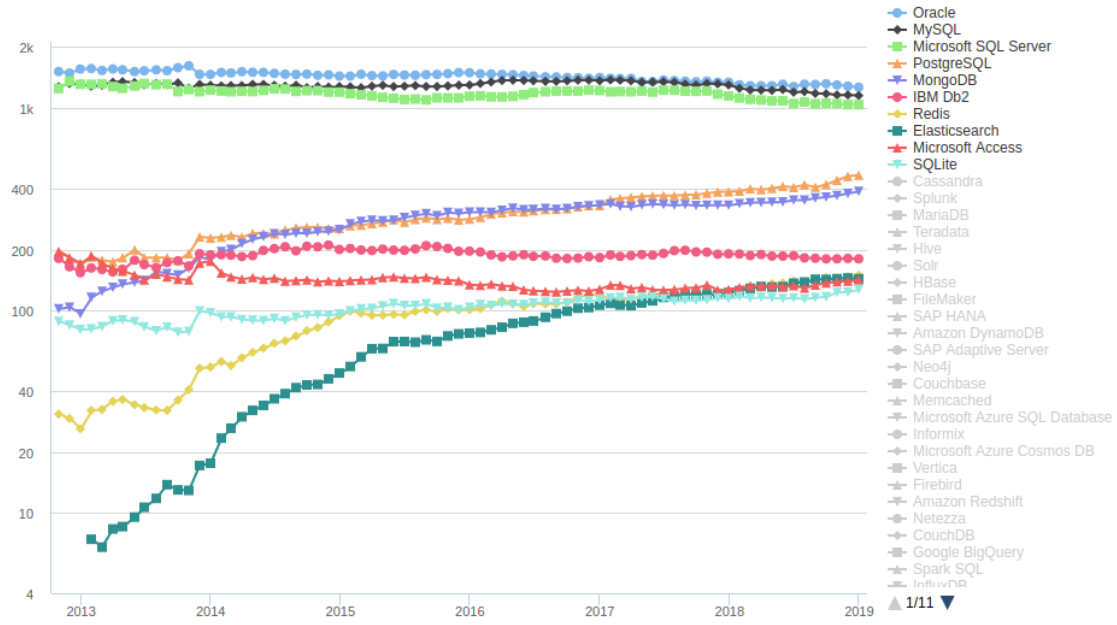

Figura 2. Comportamiento en el tiempo de la popularidad de los distintos DBMS.

Un sistema se clasifica como de software libre si el código fuente está disponible de forma gratuita y se puede usar y modificar de acuerdo con la respectiva licencia. A continuación se

muestra el comportamiento en el tiempo de la popularidad de DBMS comerciales y de software libre.

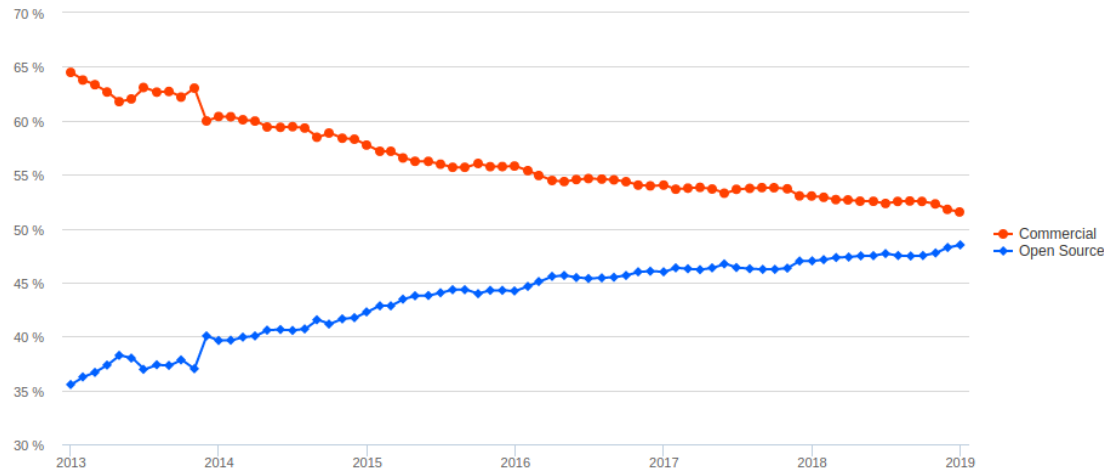

Figura 3. Comportamiento en el tiempo de la popularidad de DBMS comerciales y de software libre.

1.1.1 Relacionales

Los sistemas de base de datos relacionales (RDBMS) siguen un modelo de datos relacional (entidad-relación) orientado a tablas. El esquema de una relación o tabla se define por el nombre de la tabla y un número fijo de atributos o columnas cuyos tipos de datos son fijos. Una entidad o registro corresponde a una fila y consta de los valores de cada atributo o columna.

Sobre las relaciones o tablas de definen ciertas operaciones básicas:

- Unión, intersección y diferencia.
- Selección de un subconjunto de entidades o registros según ciertos criterios de filtro para los valores de atributo o columna de una tabla.
- Selección de un subconjunto de atributos o columnas de una tabla.

Estas operaciones básicas así como las operaciones de creación, modificación y eliminación de esquemas, las operaciones de control de transacciones y gestión de usuarios se realizan mediante lenguajes de base de datos, siendo SQL el lenguaje predominante.

Los primeros RDBMS aparecieron en el mercado a principios de la década de 1980 y desde entonces han sido los tipos de DBMS más utilizados. A continuación se muestra el comportamiento en el tiempo de la popularidad de distintos RDBMS.

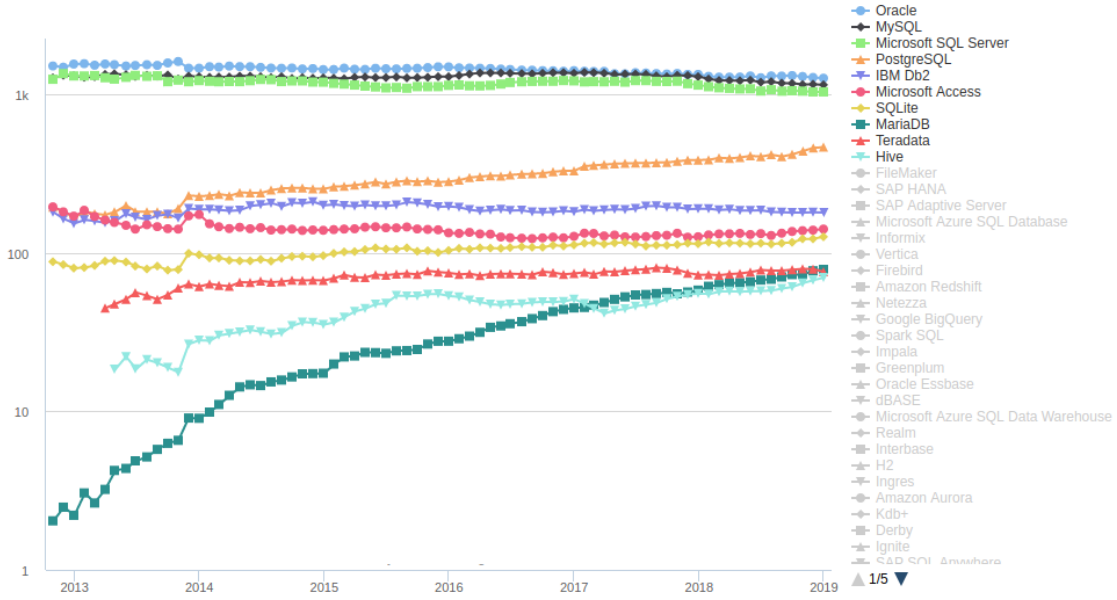

Figura 4. Comportamiento en el tiempo de la popularidad de distintos RDBMS.

A continuación se describen brevemente los más populares:

- **Oracle:** desarrollado bajo licencia comercial por Oracle. Es fácil de instalar y configurar. Está disponible en todos los sistemas operativos soportados por Oracle entre los cuales se incluye Windows, Linux y Unix. Proporciona una rápida instalación tanto en un único servidor como en un clúster. Dispone de una interfaz web que muestra el estado actual de la base de datos y del clúster y permite su administración desde cualquier navegador. El acceso a los datos se realiza a través de interfaces estándar como el lenguaje SQL o la API JDBC. Su gestión de la concurrencia asegura el máximo rendimiento para todas las cargas de trabajo. Cuando se instala en un clúster, la carga de trabajo se balancea automáticamente entre todas las máquinas disponibles, asegurando así la máxima utilización de los recursos.
- **MySQL:** desarrollado bajo licencia dual (pública general / comercial por Oracle), está considerado como el sistema de base datos de software libre más popular del mundo y, en general, uno de los más populares junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web. La versión de pago incluye productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez MySQL y Oracle. Es muy rápido en la lectura aunque se pueden producir problemas de integridad en entornos de alta concurrencia en la modificación de datos. Por lo tanto, resulta ideal en aplicaciones web donde hay baja concurrencia en la modificación y, en cambio, el entorno es intensivo en lectura de datos.
- **Microsoft SQL Server:** desarrollado bajo licencia comercial por Microsoft. Tradicionalmente ha estado disponible solo para sistemas operativos Windows pero

también lo está desde 2016 para GNU/Linux y desde 2017 para Docker. Permite trabajar en modo cliente-servidor, donde los datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a esta información. T-SQL (Transact-SQL) es el principal medio de interacción con el servidor. Permite realizar las operaciones básicas, incluyendo la creación y modificación de esquemas, la inserción y modificación de datos así como la administración del servidor como tal.

- **PostgreSQL:** desarrollado bajo licencia pública general por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por organizaciones comerciales. Su nombre hace referencia a los orígenes del proyecto como la base de datos post-Ingres. Entre sus principales características destacan una alta capacidad de concurrencia, permitiendo que mientras un proceso escribe en una tabla, otros accedan a la misma sin necesidad de bloqueos y una amplia variedad de tipos nativos además de la capacidad de crear tipos propios de datos.

1.1.2 No relacionales

Los sistemas de base de datos NoSQL son una alternativa a los RDBMS. No utilizan un modelo de datos relacional (entidad-relación) y, por lo general, no tienen una interfaz SQL.

Aunque existen desde hace muchos años, el término NoSQL se introdujo por primera vez en 2009 cuando se desarrollaron sistemas para hacer frente a los nuevos retos de los RDBMS, principalmente escalabilidad y tolerancia a fallos para grandes aplicaciones web (Big Data).

Los sistemas de base de datos NoSQL tienen una mayor flexibilidad porque usan un modelo de datos sin esquema. Esto permite una distribución más fácil de los datos en diferentes nodos, pudiendo así escalar horizontalmente (mayor rendimiento) y tener tolerancia a fallos (mayor fiabilidad). Como contrapartida suelen incumplir uno o más de los criterios ACID:

- Atomicidad: si una operación está compuesta por una serie de pasos, o todos se ejecutan o ninguno, es decir, las transacciones son completas.
- Consistencia: se ejecutan solo aquellas operaciones que no van a romper las reglas de integridad de la base de datos, es decir, cualquier operación llevará la base de datos desde un estado válido a otro también válido.
- Aislamiento: una operación no puede afectar a otras. Define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes.
- Durabilidad: una vez realizada una operación, ésta persistirá y no se podrá deshacer aunque falle el sistema, manteniendo los datos de esta forma.

Según el tipo de almacenamiento, los sistemas de base de datos NoSQL se pueden clasificar principalmente como:

- Basado en clave-valor
- Basado en documento
- Basado en columna

- Basado en grafo
- Basado en serie temporal
- Basado en motor de búsqueda

Cabe destacar que debido a la heterogeneidad de los sistemas de base de datos NoSQL, un mismo sistema puede clasificarse en distintas categorías.

La siguiente figura muestra el comportamiento en el tiempo de los cambios de popularidad de sistemas de base de datos NoSQL así como de RDBMS.

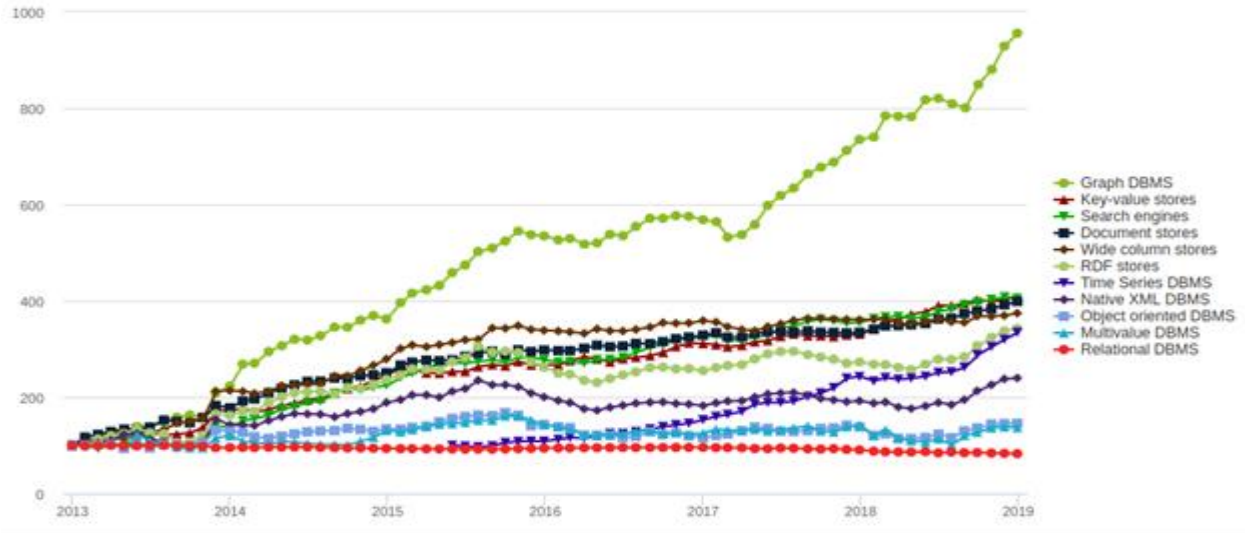

Figura 5. Comportamiento en el tiempo de los cambios de popularidad de sistemas de base de datos NoSQL así como de RDBMS.

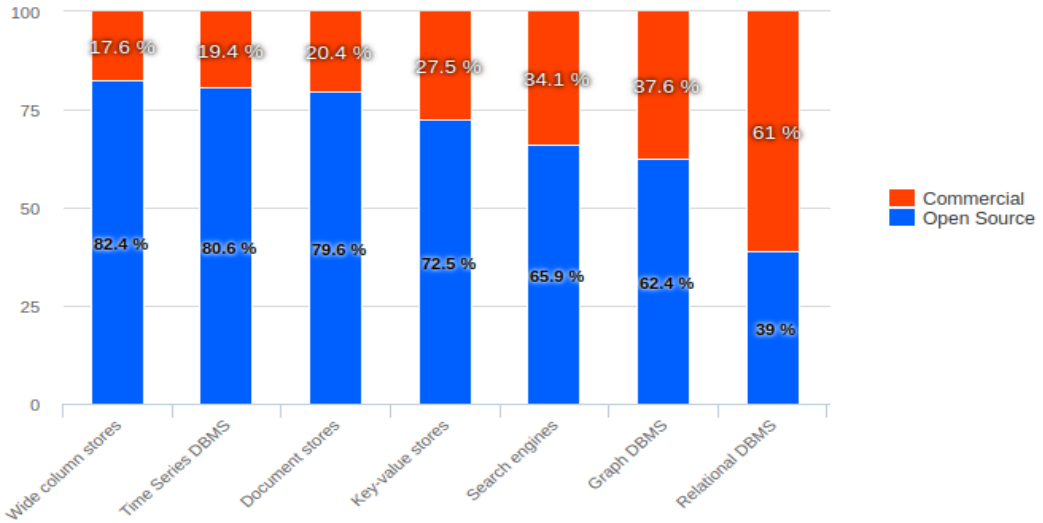

Figura 6. Popularidad por categoría de sistemas de base de datos NoSQL comerciales y de software libre.

Almacenamiento basado en clave-valor

Probablemente la forma más simple de DBMS. Solo pueden almacenar pares de clave y valor así como recuperar los respectivos valores cuando se conoce una clave determinada. Normalmente no son adecuados para aplicaciones complejas. Sin embargo, esta misma simplicidad es su principal atractivo.

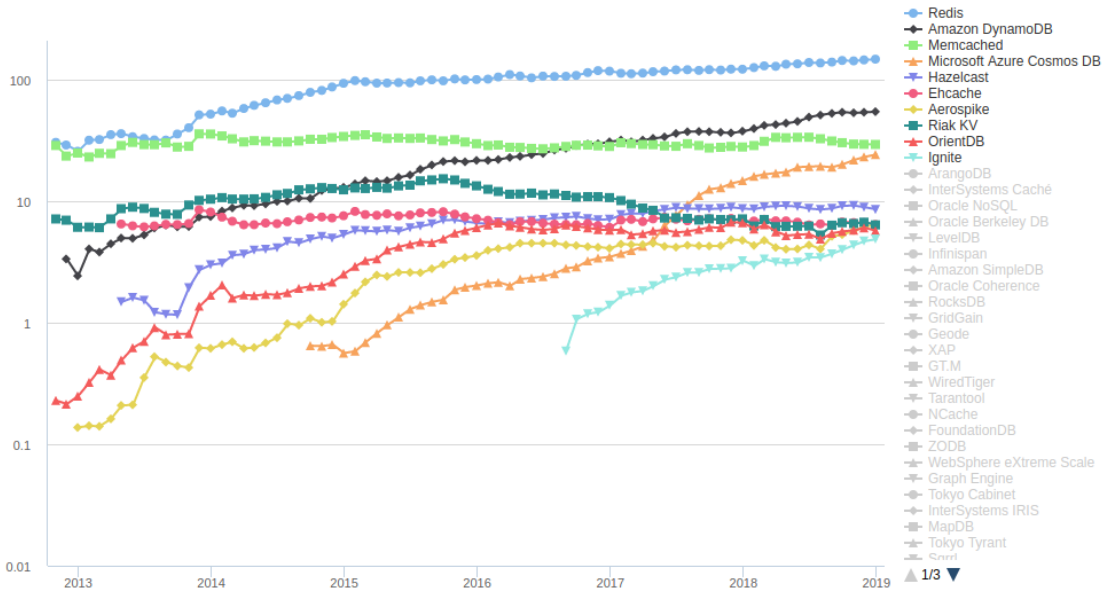

Figura 7. Popularidad de los sistemas de base de datos NoSQL con almacenamiento basado en clave-valor.

A continuación se describen brevemente los más populares:

- Redis: desarrollado bajo licencia pública general, es un sistema de base de datos en memoria, basado en el almacenamiento clave-valor. Su modelo de datos se basa en una estructura del tipo diccionario o tabla hash que relaciona una llave con un contenido almacenado en un índice. La información se guarda normalmente en memoria RAM aunque también se pueden persistir los datos de dos formas a costa de reducir el rendimiento. La primera consiste en realizar distintos snapshots del contenido en memoria cada cierto tiempo. Sin embargo, esta persistencia no sería realmente durable ya que las capturas son asíncronas. La segunda consiste en escribir sobre un archivo al que sólo se puede añadir información cada modificación que se realice sobre los datos en memoria, pudiendo regenerar posteriormente dichos datos. Usa una replicación de tipo maestro-esclavo, es decir, los datos de un servidor se pueden replicar en muchos esclavos (un esclavo puede ser también maestro de otro esclavo). Esta replicación permite escalar la lectura (no así la escritura) y redundar los datos aunque puede ocasionar inconsistencias no intencionales en los datos.
- Amazon DynamoDB: servicio de base de datos NoSQL ofrecido por Amazon Web Services. Mediante una replicación síncrona en múltiples centros de datos, proporciona una alta durabilidad y disponibilidad. Difiere de otros servicios de Amazon

al permitir comprar un servicio basado en el rendimiento y no en el almacenamiento. Dispone de la capacidad de escalar automáticamente. Ofrece integración con Hadoop a través de Elastic MapReduce. Unas métricas de rendimiento ayudan a aprovisionarlo correctamente.

- Memcached: desarrollado para el almacenamiento en caché de datos. Permite reducir las necesidades de acceder a un origen de datos externo como un sistema de base de datos o una API. Tiene versiones para Linux, Windows y MacOS y se distribuye bajo licencia de software libre. Su funcionamiento se basa en una tabla hash distribuida. Conforme ésta se va llenando, los datos que más tiempo llevan sin ser utilizados se borran para dar espacio a los nuevos. Las aplicaciones normalmente comprueban primero si pueden acceder a los datos a través de Memcached antes de recurrir a un almacén de datos más lento. Tiene una arquitectura cliente-servidor donde los servidores mantienen un array asociativo clave-valor y los clientes leen o escriben sobre dicho array. Cada cliente mantiene una lista de todos los servidores. Si un cliente desea leer o escribir el valor correspondiente a cierta clave, primero realiza un cálculo mediante un algoritmo de hash para determinar el servidor que va a utilizar. Después se pone en contacto con este servidor que, a su vez, usará otro hash para determinar dónde leer o escribir dicho valor.
- Microsoft Azure Cosmos DB: servicio de base de datos NoSQL multimodelo de Microsoft. Es independiente del esquema y horizontalmente escalable. Internamente almacena artículos en contenedores. Estos dos conceptos, artículos y contenedores, dependen de la API utilizada. Por ejemplo, los artículos serían documentos y los contenedores colecciones cuando se utiliza la API compatible con MongoDB). El modelo de datos interno se expone a través de una API de SQL propietaria que permite crear, actualizar y eliminar contenedores y artículos. Adicionalmente, dispone de otras cuatro API compatibles con los protocolos de conexión de MongoDB (documento), Gremlin (grafo), Cassandra (clave-valor) y Azure Table Storage (columna). Dispone de la capacidad de partición automática. Los contenedores particionados abarcan varias particiones físicas con artículos distribuidos por una clave de partición. El número de particiones dependerá del tamaño y las necesidades de rendimiento. Reserva los recursos necesarios para garantizar el rendimiento solicitado al tiempo que mantiene la latencia de la solicitud por debajo de 10 ms tanto para lecturas como escrituras.

Almacenamiento basado en documento

Se caracterizan por una organización de los datos sin esquemas. Los registros no necesitan tener una estructura uniforme, es decir, diferentes registros pueden tener diferentes columnas y los tipos de valores de columnas individuales pueden ser diferentes para cada registro. Además, los registros pueden tener una estructura anidada y las columnas más de un valor. Comparado con un RDBMS, las colecciones son como tablas y los documentos son como registros de una tabla. La diferencia es que cada registro de una tabla tiene la misma cantidad

de columnas mientras que cada documento en una colección puede tener diferentes campos. En un documento se pueden agregar, eliminar, modificar o renombrar nuevos campos en cualquier momento ya que no hay un esquema predefinido. La estructura de un documento es simple y está compuesta por pares clave-valor.

A menudo utilizan notaciones internas, principalmente JSON. Estos documentos JSON también podrían almacenarse como texto puro en bases de datos NoSQL con almacenamiento basado en clave-valor o RDBMS. Sin embargo, eso requeriría en el lado del cliente del procesamiento de dichas estructuras, además de no disponer de las características ofrecidas por las bases de datos NoSQL con almacenamiento basado en documento como, por ejemplo, los índices secundarios.

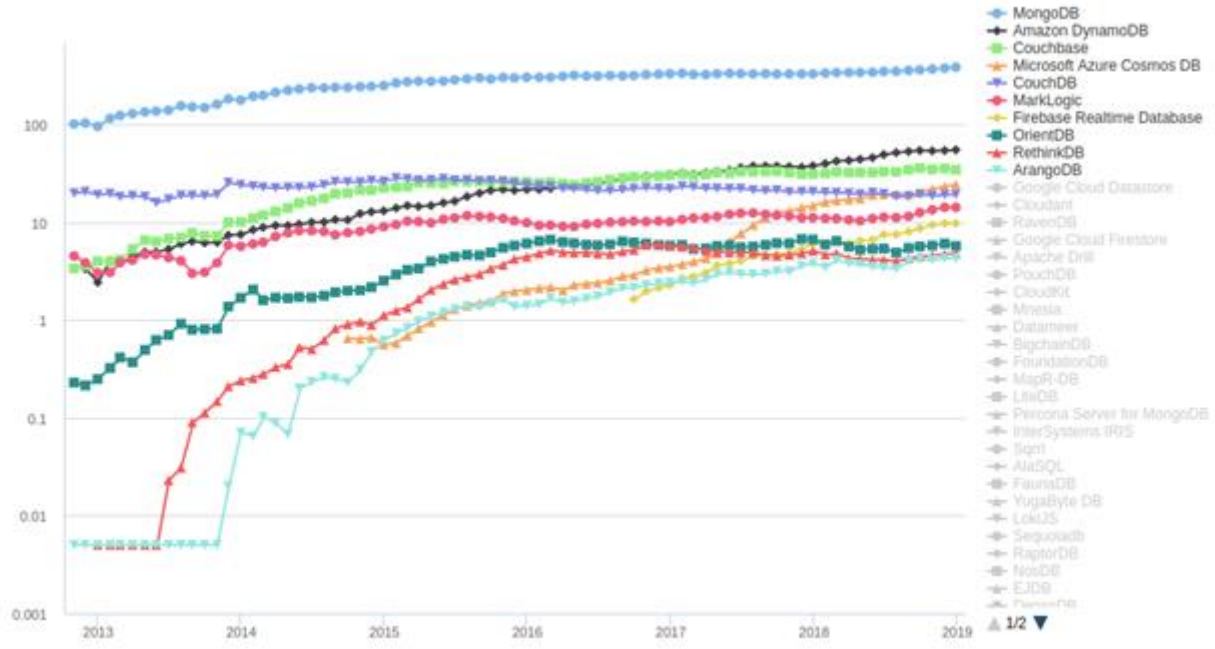

Figura 8. Popularidad de los sistemas de base de datos NoSQL con almacenamiento basado en documento.

A continuación se describe brevemente la más popular:

- MongoDB: desarrollado bajo licencia pública general, está disponible para los sistemas operativos Windows, Linux, OS X y Solaris. Guarda la estructura de los datos en documentos BSON mediante un esquema dinámico. Soporta el tipo de replicación primario-secundario. El primario puede ejecutar comandos de lectura y escritura mientras que los secundarios replican los datos del primario y sólo se pueden usar para lectura o copia de seguridad. Los secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder. Cada grupo de primario y secundarios se denomina conjunto de replica. También puede escalar horizontalmente mediante el concepto de fragmento. A partir de una clave de fragmentación se distribuyen los datos de una colección en múltiples fragmentos,

pudiendo constituir cada fragmento un conjunto de réplica. Solo implementa las propiedades ACID dentro de un mismo documento.

Almacenamiento basado en columna

Los datos se almacenan en registros con la capacidad de contener un gran número de columnas dinámicas. A diferencia de un RDBMS, cada registro (pensar en una fila de un RDBMS) no requiere un único valor por columna sino que es posible modelar una familia de columnas. Cada una de estas familias puede constar de varios campos. Además, tampoco requiere que los campos estén siempre presentes ni tener que rellenarlos con un valor nulo si no lo estuvieran como sí lo requiere un RDBMS, evitando así el problema de datos dispersos y preservando espacio en el disco.

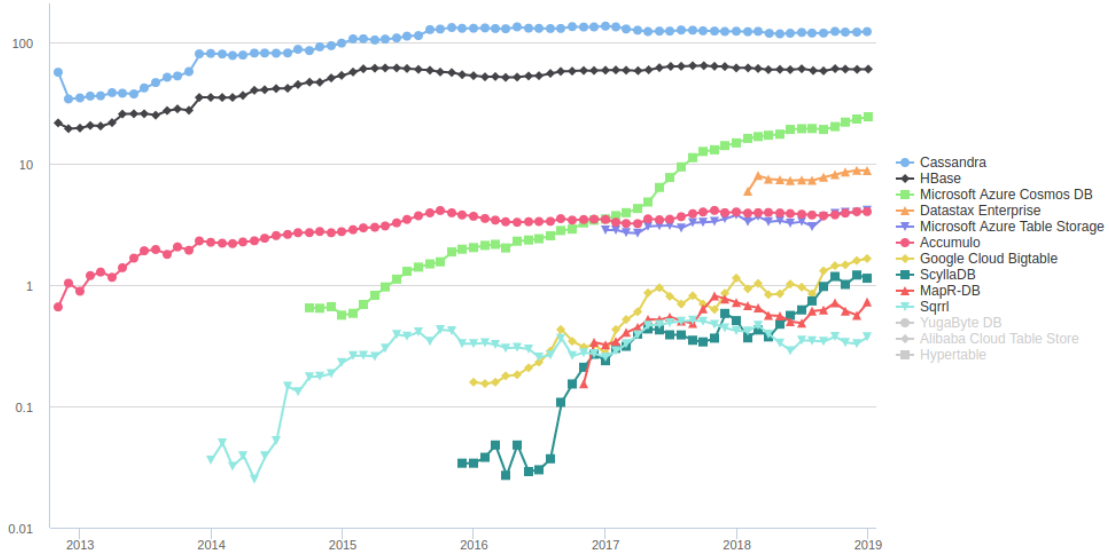

Figura 9. Popularidad de los sistemas de base de datos NoSQL con almacenamiento basado en columna.

A continuación se describen brevemente las más populares:

- Cassandra: desarrollada bajo licencia pública general, es un híbrido entre una base de datos NoSQL con almacenamiento basado tanto en columna como en clave-valor. Su objetivo principal es la escalabilidad lineal y la disponibilidad. Su arquitectura distribuida está basada en una serie de nodos iguales que se comunican mediante un protocolo P2P. Todos los nodos tienen el mismo rol por lo que no hay un único punto de fallo y cada nodo puede dar servicio a cualquier solicitud. Los datos se replican automáticamente en distintos nodos para recuperarse frente a posibles fallos. Utiliza un API propia para acceder a llos datos aunque actualmente está apostando por un lenguaje denominado CQL (Cassandra Query Language) que tiene una sintaxis similar a SQL aunque con muchas menos funcionalidades.
- Hbase: desarrollado bajo licencia pública general, forma parte del proyecto Hadoop y se ejecuta sobre HDFS (sistema de archivos distribuidos de Hadoop). Permite acelerar

operaciones de lectura y escritura sobre grandes conjuntos de datos con un alto rendimiento y una baja latencia de entrada/salida. El proyecto Phoenix proporciona una capa de SQL para Hbase así como acceso vía JDBC, pudiendo integrarse con diversas herramientas de analítica de datos e inteligencia de negocio. Análogamente, el proyecto Trafodion proporciona un motor de consulta SQL con drivers ODBC y JDBC y protección de transacciones ACID utilizando HBase como motor de almacenamiento.

Almacenamiento basado en grafo

Representan los datos mediante una estructura gráfica compuesta por nodos que permite definir relaciones entre ellos. De esta forma pueden procesar fácilmente los datos y calcular propiedades específicas del grafo como, por ejemplo, la cantidad de pasos necesarios para ir de un nodo a otro.

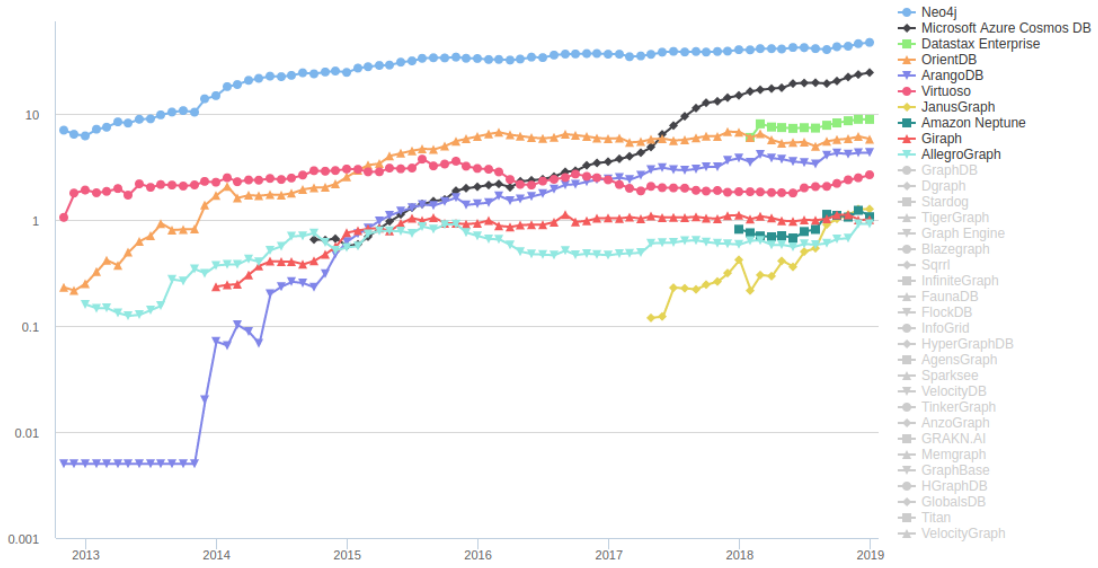

Figura 10. Popularidad de los sistemas de base de datos NoSQL con almacenamiento basado en grafo.

A continuación se describe brevemente la más popular:

- Neo4j: desarrollado bajo licencia dual (pública general / comercial), es un motor de persistencia embebido, basado en disco, completamente transaccional que almacena datos estructurados en grafos en lugar de tablas.

Almacenamiento basado en serie temporal

Están optimizados para manejar datos de series temporales, donde cada muestra está asociada a una marca de tiempo. Permiten recopilar, almacenar y consultar de manera eficiente varias series temporales con un alto volumen de transacciones. Si bien estos datos se pueden gestionar también con otros DBMS, desde sistemas de base de datos NoSQL con

almacenamiento basado en clave-valor hasta RDBMS, los desafíos específicos a menudo requieren de estos sistemas especializados.

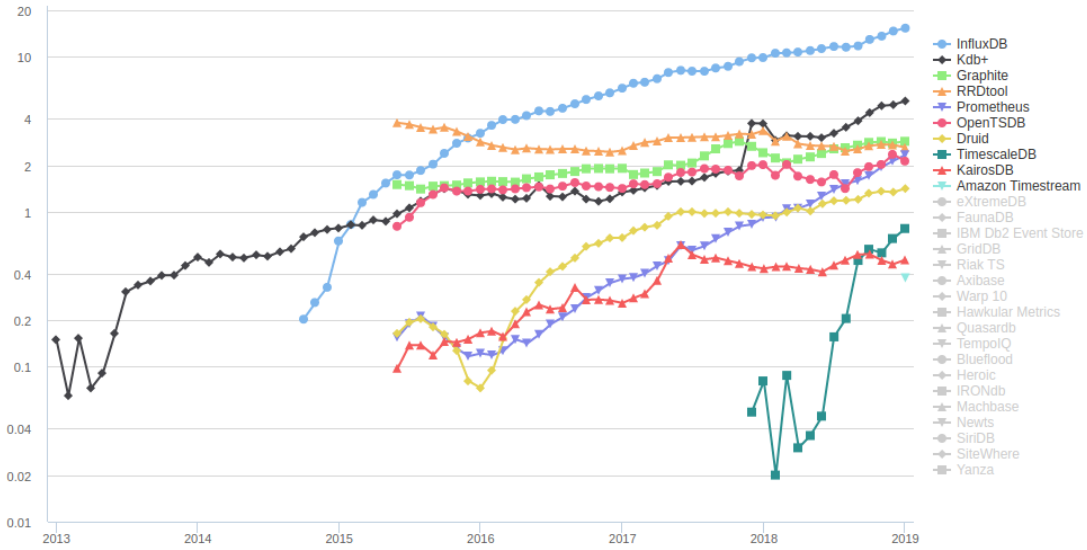

Figura 11. Popularidad de los sistemas de base de datos NoSQL con almacenamiento basado en serie temporal.

A continuación se describen brevemente las más populares:

- InfluxDB: desarrollado bajo licencia dual (pública general / comercial), está optimizada para el almacenamiento rápido y la alta disponibilidad de datos de series temporales. Las políticas de retención controlan cómo se reducen los muestreos y se eliminan los datos. Acepta datos a través de HTTP, TCP y UDP. En 2016 se anunció que la capacidad de escalabilidad horizontal solo estaría disponible en la versión de pago.
- Kdb+: desarrollado bajo licencia comercial por Kx Systems y disponible en los sistemas operativos Linux, Windows y Solaris, se usa comúnmente para almacenar, analizar, procesar y recuperar grandes conjuntos de datos a alta velocidad. Dispone de capacidades en memoria. Usa el lenguaje q para agregar y analizar los datos, realizar funciones estadísticas y unir conjuntos de datos, además de admitir consultas SQL.

Almacenamiento basado en motor de búsqueda

Están dedicados a la búsqueda de contenido, proporcionando principalmente las siguientes características:

- Soporte para expresiones de búsqueda complejas.
- Reducción de las palabras a su raíz.
- Clasificación y agrupación de los resultados de búsqueda.
- Búsqueda geoespacial.
- Búsqueda distribuida de alta escalabilidad.

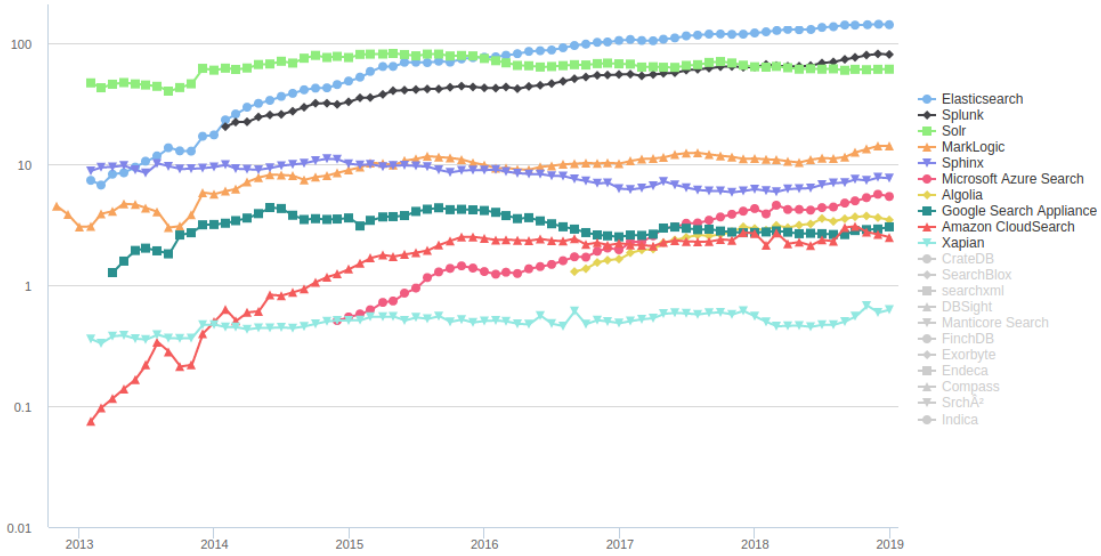

Figura 12. Popularidad de los sistemas de base de datos NoSQL con almacenamiento basado en motor de búsqueda.

A continuación se describen brevemente las más populares:

- Elasticsearch: desarrollada bajo licencia pública general, es un motor de búsqueda de texto basado en Lucene. Utiliza un lenguaje sumamente flexible y de gran alcance, además de simple, que permite conocer y explorar los datos de la mejor manera. Al ser distribuido es fácilmente escalable. Las respuestas se devuelven en formato JSON, casi en tiempo real.
- Splunk: desarrollado bajo licencia dual (pública general / comercial), permite buscar, monitorizar y analizar datos procedentes de aplicaciones, sistemas e infraestructuras IT a través de una interfaz web. Los datos primeramente se capturan, indexan y correlan en tiempo real y posteriormente se almacenan en un repositorio. Permite crear fácilmente gráficos, alertas y paneles. La versión gratuita está limitada a 500 MB de datos al día y carece de algunas de las funcionalidades de la versión de pago.
- Solr: desarrollado bajo licencia pública general, es un motor de búsqueda basado también en Lucene. Permite la búsqueda por facetas o navegación facetada. Esta técnica permite acceder a información organizada de acuerdo a un sistema de clasificación de facetas donde los usuarios pueden explorar una colección de datos aplicando una serie filtros.

1.2 Sistemas de procesamiento

Los sistemas de procesamiento realizan la computación de los datos. Estos se leen de un medio de almacenamiento no volátil o a medida que se van ingiriendo en el sistema. Algunos sistemas procesan los datos por lotes, otros los procesan en tiempo real y otros pueden procesarlos de ambas formas.

1.2.1 Por lotes

El procesamiento por lotes opera sobre un conjunto estático de datos. Este conjunto constituye una colección finita de datos respaldados generalmente por algún tipo de almacenamiento permanente. Resulta adecuado para cálculos donde se requiera el acceso al conjunto completo de los datos.

Las tareas que requieren grandes volúmenes de datos a menudo se manejan mejor mediante operaciones por lotes. Éstas se usan habitualmente sobre datos históricos. La contraprestación por el manejo de tales volúmenes es un tiempo de cálculo más prolongado. Debido a esto, el procesamiento por lotes no es apropiado en situaciones donde el tiempo de procesamiento es especialmente importante.

Hadoop es un sistema de procesamiento por lotes escalable de software libre. MapReduce es su paradigma de procesamiento. Resulta adecuado para procesar conjuntos de datos muy grandes donde el tiempo no es un factor importante. Su compatibilidad y capacidad de integración con otros sistemas de procesamiento hacen de él la base para múltiples cargas de trabajo de procesamiento mediante tecnologías diversas. A continuación se describen brevemente sus componentes principales:

- HDFS: sistema de archivos distribuido que coordina el almacenamiento y la replicación de los datos en los distintos nodos del clúster de Hadoop. Se utiliza tanto como fuente de datos como almacenamiento de los resultados intermedios y finales del procesamiento.
- YARN: responsable de coordinar y administrar los recursos subyacentes y programar los trabajos a ejecutar.
- MapReduce: paradigma de procesamiento por lotes basado en un algoritmo de mapeo y reducción de pares clave-valor. Primeramente se lee el conjunto de datos de HDFS y se divide en distintos fragmentos que se distribuyen entre los nodos disponibles del clúster de Hadoop (mapeo). Posteriormente se realiza el procesamiento en cada uno de los nodos de su respectivo subconjunto de datos. Los resultados intermedios se almacenan en HDFS, agrupándose y volviendo a distribuirse de acuerdo a su clave. Finalmente se combinan estos resultados (reducción) y se vuelven a almacenar en HDFS.

Debido a que esta metodología se basa en el almacenamiento permanente, realizando múltiples lecturas y escrituras, resulta bastante lenta. Sin embargo, como el espacio en disco es normalmente uno de los recursos más abundantes, MapReduce puede manejar enormes conjuntos de datos. Además puede ejecutarse sobre hardware menos costoso porque no trata de almacenar todos los datos en memoria. MapReduce es altamente escalable, habiéndose utilizado en producción sobre decenas de miles de nodos. Muchos otros sistemas de procesamiento tienen integraciones con Hadoop para usar HDFS y YARN.

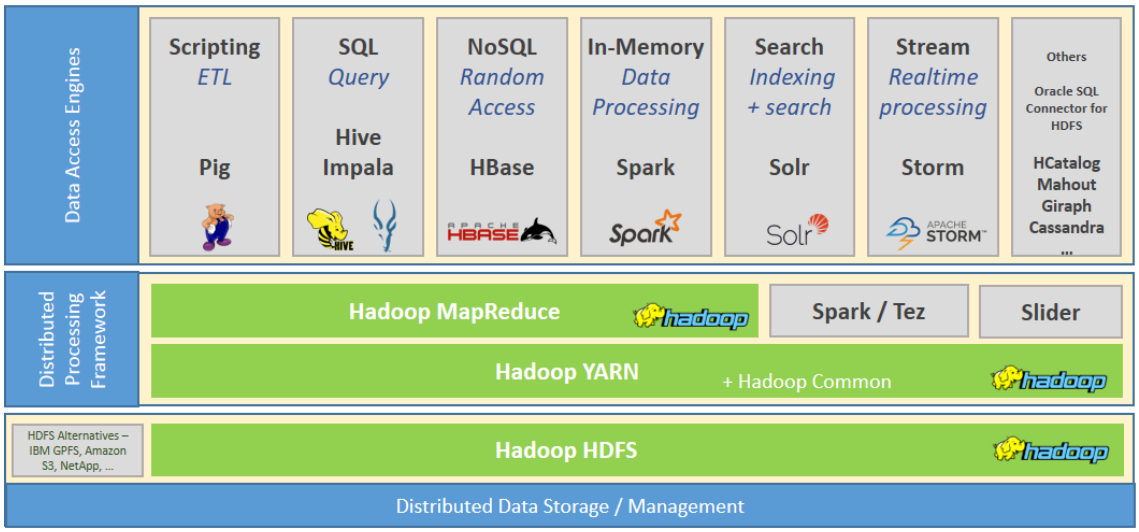

Figura 13. Arquitectura de Hadoop.

1.2.2 En tiempo real

Los sistemas de procesamiento en tiempo real procesan los datos a medida que ingresan al sistema. Esto requiere un modelo diferente al paradigma de procesamiento por lotes. En lugar de definir las operaciones que se aplican al conjunto total de datos, se definen las operaciones que se aplican a cada elemento individual de los datos a medida que ingresa en el sistema. El procesamiento está basado en eventos y no finaliza hasta que se detiene explícitamente. Los resultados estarán disponibles de inmediato y se actualizarán continuamente a medida que lleguen nuevos datos. Estos sistemas de procesamiento pueden manejar una cantidad de datos casi ilimitada pero solo procesan uno (procesamiento en tiempo real estricto) o muy pocos (procesamiento por microlotes) a la vez.

Storm es un sistema de procesamiento en tiempo real (estricto) diseñado para una latencia extremadamente baja. Garantiza que cada mensaje se procesa al menos una vez aunque pudiera haber duplicados. Tampoco garantiza que los mensajes sean procesados en orden. En términos de interoperabilidad se puede integrar con el administrador de recursos YARN de Hadoop, facilitando así su conexión a un componente existente del ecosistema Hadoop. Además puede utilizarse con una gran cantidad de lenguajes de programación.

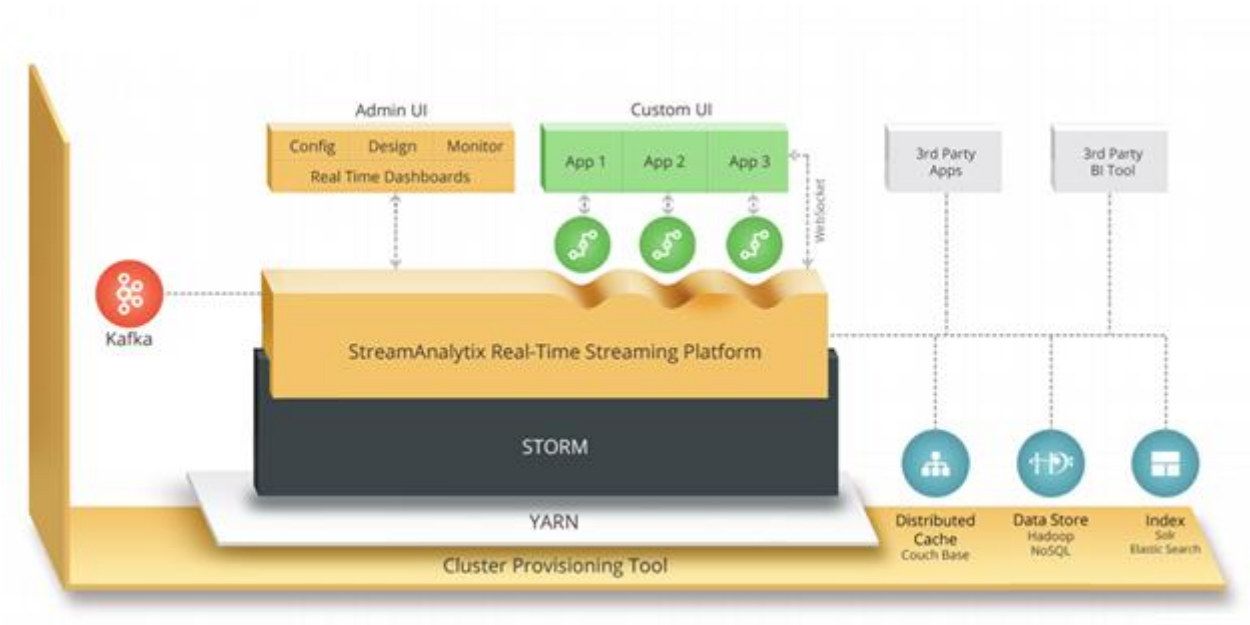

Figura 14. Arquitectura de Storm.

Samza es un sistema de procesamiento por microlotes estrechamente vinculado a Kafka. Si bien Kafka puede ser utilizado por muchos sistemas de procesamiento en tiempo real, Samza está diseñado específicamente para aprovechar la arquitectura y las garantías únicas que proporciona Kafka. Por ejemplo, Kafka ofrece almacenamiento replicado de los datos a los que se puede acceder con baja latencia. También proporciona un modelo de multisubscriptor para cada partición. Todos los resultados, incluidos los intermedios, se escriben en Kafka, pudiendo ser consumidos independientemente por etapas posteriores. Utiliza el YARN de Hadoop para la administración de los recursos.

A continuación se describen brevemente los componentes principales de Kafka:

- Topics: flujo de información a la que los consumidores se pueden suscribir.
- Particiones: para distribuir un topic entre los nodos de Kafka, los mensajes entrantes se dividen en distintas particiones. Estas particiones se basan en una clave determinada, de tal forma que cada mensaje con la misma clave se enviará a la misma partición. Además, las particiones respetan el orden de llegada de los mensajes.
- Brokers: nodos individuales que conforman el clúster de Kafka.
- Productores: cualquier componente que escriba en un topic. El productor proporciona la clave utilizada para particionar dicho topic.
- Consumidores: cualquier componente que lee de un topic. Los consumidores son responsables de mantener la información sobre su propio offset, de modo que estén al tanto de qué mensajes se han procesado si ocurriera un error.

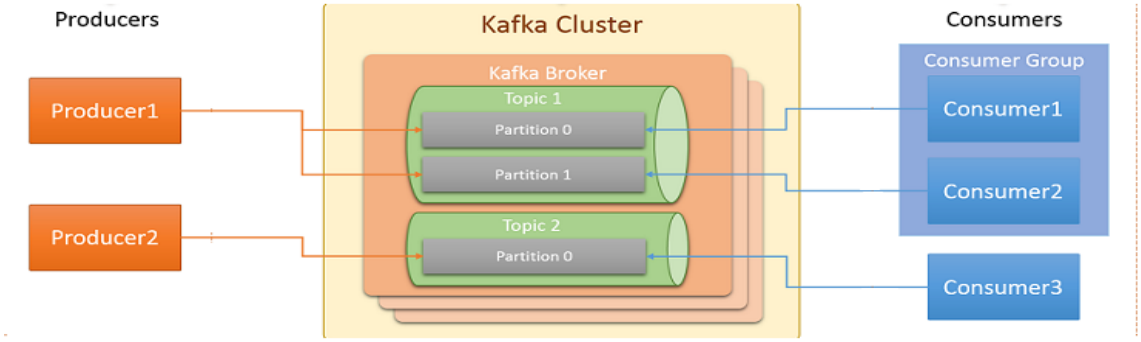

Figura 15. Arquitectura de Kafka.

Como Kafka representa un registro inmutable, los flujos en Samza son inmutables. Es decir, cualquier transformación crea un nuevo flujo de datos que puede ser consumido sin afectar el flujo inicial. Escribir directamente en Kafka también elimina los problemas de cuello de botella. Estos se producen cuando los picos de carga causan una afluencia de datos a una tasa mayor de la que los componentes pueden procesar en tiempo real, atascando así el procesamiento y pudiendo incluso perder datos. Kafka está diseñado para almacenar los datos durante largos períodos de tiempo, lo que significa que los flujos de datos pueden procesarse a conveniencia, pudiendo reiniciarse sin consecuencias.

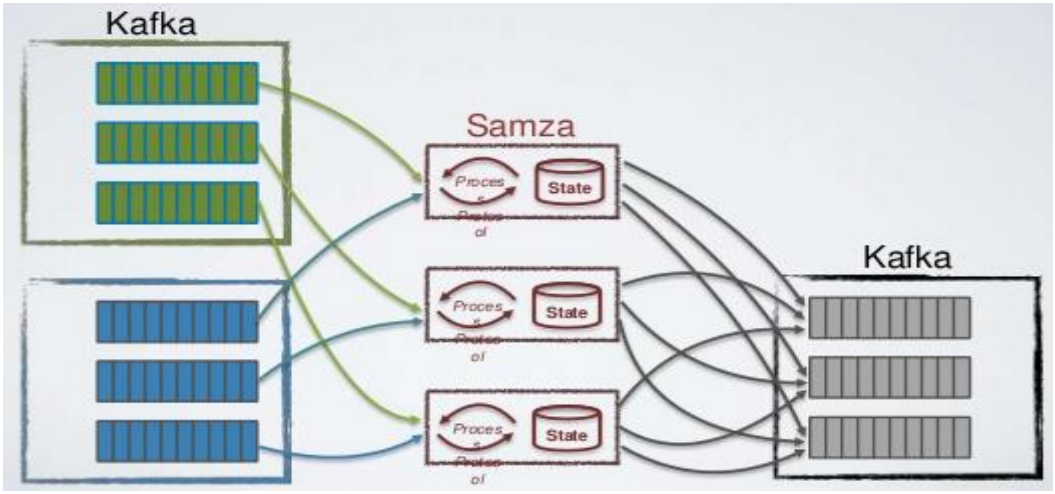

Figura 16. Arquitectura de Samza.

Algunos sistemas de procesamiento pueden manejar cargas de trabajo tanto por lotes como en tiempo real, pudiendo utilizar los mismos componentes y APIs para ambos tipos de datos. Si bien los sistemas centrados en un tipo de procesamiento constituyen una solución óptima para determinados casos de uso, los sistemas híbridos intentan ofrecer una solución general para el procesamiento de datos.

Spark es un sistema de procesamiento por lotes que cuenta también con capacidades de procesamiento en tiempo real. Basado en muchos de los mismos principios del paradigma MapReduce de Hadoop, está diseñado para acelerar las cargas de trabajo de procesamiento por lotes realizándolas en memoria. A diferencia de MapReduce, Spark procesa todos los datos en memoria, solo interactuando con la capa de almacenamiento para cargar inicialmente los datos en memoria y finalmente para almacenar los resultados finales. Todos los resultados intermedios se almacenan también en memoria. Se puede ejecutar como un clúster independiente o conectarse a Hadoop, sustituyendo al paradigma MapReduce.

Si bien el procesamiento en memoria contribuye sustancialmente al incremento de la velocidad, Spark también es más rápido que MapReduce en tareas relacionadas con el disco debido a la optimización holística que se puede lograr analizando con anticipación el conjunto completo de tareas. Esto se logra mediante grafos acíclicos dirigidos (DAG) que representan todas las operaciones que se deben realizar, los datos a utilizar y las relaciones entre ellos, permitiendo a la CPU una mayor capacidad para coordinar y ejecutar el trabajo.

Para realizar el procesamiento por lotes en memoria, Spark usa unas estructuras de datos que se denotan como RDD. Son estructuras inmutables residentes en memoria que representan colecciones de datos. Las operaciones sobre RDDs producen nuevos RDDs y cada RDD puede rastrear su linaje a través de los distintos RDDs a partir de los cuales se obtuvo y, en última instancia, hasta los datos en el disco. Mediante los RDDs, Spark garantiza la tolerancia a fallos sin necesidad de volver a escribir en el disco después de cada operación.

Para lidiar con la disparidad entre el diseño del sistema de procesamiento por lotes y las características de las cargas de trabajo en tiempo real, Spark Streaming implementa una estrategia para tratar los flujos de datos como una serie de lotes muy pequeños (microlotes) que se pueden manejar mediante la semántica propia de Spark. La adaptación de la metodología por lotes para el procesamiento en tiempo real implica el almacenamiento intermedio de los datos a medida que ingresan en el sistema.

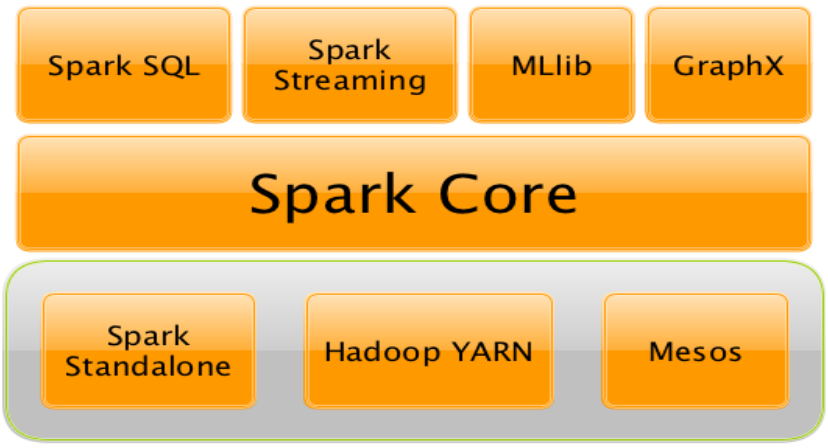

Figura 17. Arquitectura de Spark.

Flink es un sistema de procesamiento en tiempo real que también puede manejar tareas por lotes. Considera que los lotes son simplemente flujos de datos con límites finitos y, por lo tanto, trata el procesamiento por lotes como un subconjunto del procesamiento en tiempo real.

A continuación se describen brevemente los componentes principales de Flink:

- Flujos: conjuntos de datos inmutables e ilimitados que fluyen a través del sistema.
- Operadores: funciones que operan sobre flujos de datos para producir otros flujos.
- Fuentes: punto de entrada para los flujos que ingresan en el sistema.
- Sumideros: lugar donde los flujos salen del sistema.

Por razones de rendimiento, Flink administra su propia memoria en lugar de depender de los mecanismos nativos de recolección de basura de Java. Además, se integra fácilmente con YARN, HDFS y Kafka y su compatibilidad con los programas nativos de Storm y Hadoop le permite ejecutar tareas escritas para estos sistemas de procesamiento.

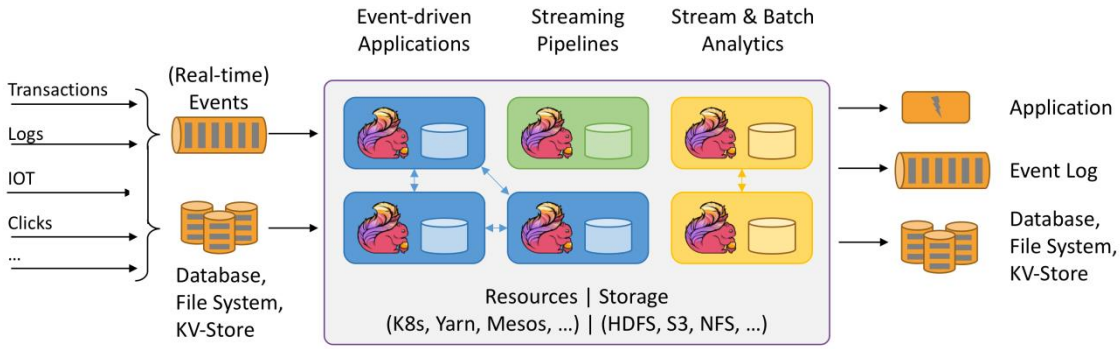

Figura 18. Arquitectura de Flink.

1.3 Sistemas de inteligencia de negocio

En las actividades diarias de cualquier organización se genera una gran cantidad de datos. El análisis de estos datos permite optimizar la toma de decisiones. Sin embargo, este análisis es muy difícil de hacer mediante los sistemas transaccionales tradicionales (OnLine Transaction Processing, OLTP) que están orientados al registro de transacciones (operaciones de lectura, escritura, borrado y modificación) pero no a la realización de consultas.

Para solucionar este problema se introduce la Inteligencia de Negocio (Business Intelligence, BI) basada en sistemas OLAP (OnLine Analytical Processing) que están orientados a la realización de consultas, procesando grandes volúmenes de datos procedentes de distintas fuentes para así extraer información útil. El acceso a los datos suele ser sólo de lectura, con valores agregados precalculados para realizar las consultas más rápidamente. Además, los datos no se suelen actualizar en tiempo real sino periódicamente por lotes.

Todas las empresas almacenan grandes volúmenes de datos de sus operaciones diarias pero estos no dejan de ser sino la representación alfanumérica de un atributo o hecho, sin ningún tipo de significado por sí mismo. La información consiste en darle sentido a estos datos en un contexto determinado. Una solución de BI nos permite observar y comprender nuestro entorno, qué está ocurriendo y por qué, predecir qué ocurrirá y decidir sobre qué camino se ha de seguir. Da una visión completa del negocio así como las pautas que han de seguirse en el futuro.

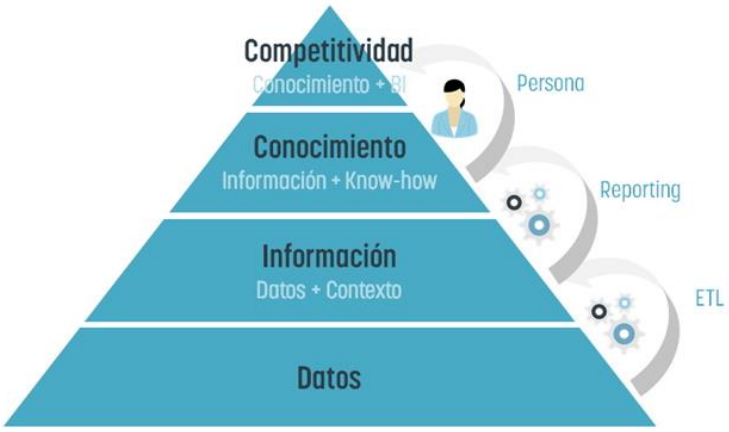

Figura 19. Pirámide del conocimiento.

Atendiendo a la arquitectura general de una aplicación de BI, el esquema básico de un proyecto de BI se describe en la siguiente figura:

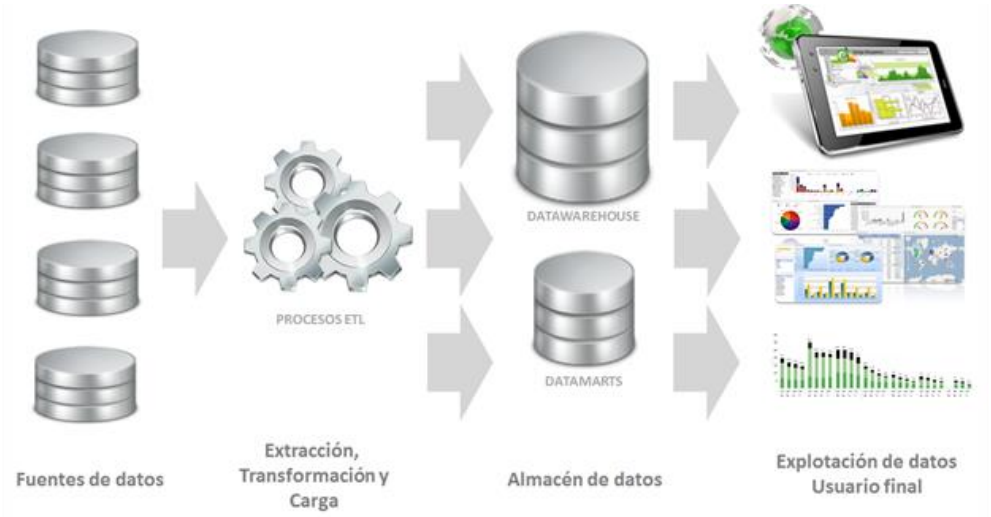

Figura 20. Esquema básico de un sistema de BI.

Primeramente se realiza la extracción de los datos de las distintas fuentes. Posteriormente se transforman en base a las preguntas a las que se quiera dar respuesta y se cargan en un almacén de datos. Finalmente, las herramientas de explotación y visualización de datos

propician su análisis mediante informes y cuadros de mando para dar respuesta a las diferentes preguntas.

A continuación se describen brevemente algunas de las fuentes de datos más utilizadas en la actualidad:

- Bases de datos relacionales: la información se organiza en tablas relacionadas entre sí mediante identificadores. Cumplen con el modelo ACID que garantiza la atomicidad, la consistencia, el aislamiento y la durabilidad de las operaciones.
- Bases de datos no relacionales: la información se organiza normalmente en documentos sin un identificador que sirva de relación entre un conjunto y otro de datos. Muy útiles cuando no hay un esquema exacto de lo que se va a almacenar.
- Archivos CSV: archivos de texto que permiten representar la información de una tabla donde los valores están separados por comas u otros separadores.
- Archivos de hoja de cálculo: las hojas de cálculo son una de las herramientas más usadas para procesar y analizar datos. Fáciles de usar debido a su estructura de filas, columnas y celdas bastante parecida a una tabla.

Los procesos de ETL proporcionan utilidad al conjunto de datos iniciales, permitiendo obtener información de los mismos. A continuación se describen brevemente:

- Extracción: obtención de los datos de las distintas fuentes, ya sean bases de datos relacionales o no relacionales, archivos, etc. Debe causar el menor impacto en los sistemas origen. Si los datos a extraer fueran muchos, estos sistemas podrían ralentizarse e incluso colapsarse, provocando perdida de información. Por ese motivo, las operaciones de extracción se suelen programar en horarios o días donde el impacto es mínimo.
- Transformación: realización de los cálculos necesarios o adecuación del formato de los datos extraídos para su posterior explotación.
- Carga: volcado de los datos procedentes de la fase de transformación en el almacén de datos con el objetivo de analizarlos para apoyar la toma de decisiones.

Figura 21. ETL.

El nacimiento y evolución de los procesos de ETL va de la mano del almacén de datos. Las empresas necesitaban integrar los datos desde sus diferentes repositorios origen y cargarlos en un único repositorio destino. Originalmente, estos procesos se desarrollaban en lenguajes

de programación clásicos, siendo SQL el que más se asentó en el mercado. A medida que los almacenes de datos fueron ganando importancia en las grandes corporaciones, la programación de estos procesos de ETL comenzó a tener un número elevado de líneas de código que los hacía muy difíciles de mantener. La dificultad de su mantenimiento y su lenta curva de aprendizaje propició la búsqueda de alternativas.

A mediados de los 90, las empresas más punteras dentro del mundo de los sistemas de información decidieron invertir en desarrollar sus propias herramientas. Empresas como IBM, Oracle o SAS comienzan a lanzar potentes herramientas orientadas al diseño y desarrollo de procesos de ETL sin la necesidad de tener que programarlos exclusivamente en código. Así nacen IBM DataStage, Oracle Data Integrator y SAS Data Integration.

Estas herramientas destacan por su fiabilidad pero también por el alto coste de sus licencias, limitando su nicho de mercado a las grandes empresas. Sin embargo, el crecimiento de los sistemas de BI dentro de compañías no tan grandes hizo que las empresas dedicadas al software de libre centrasen su atención en los procesos de ETL. Algunos ejemplos son Talend, Jaspersoft ETL o la herramienta de software libre por excelencia, Pentaho Data Integration.

Un almacén de datos es un conjunto de información extraída de diferentes fuentes de datos tanto internas como externas (archivos planos, hojas de cálculo, bases de datos, etc). Este almacén de datos puede ser un único gran contenedor (Data Warehouse, DW) o pequeños contenedores (Data Marts, DM). Ambos comparten aspectos técnicos y características generales pero los DM se diseñan orientados a fines departamentales (optimizados para áreas concretas). A continuación se describen brevemente sus principales características:

- Orientado a temas: en los sistemas transaccionales, la base de datos se optimiza para ejecutar tareas operacionales como agregar, modificar, eliminar, mover y copiar información. Por el contrario, en los sistemas dimensionales, la base de datos se optimiza para analizar temas más orientados al modelo de negocio de una empresa.
- Integrado: permite que los datos tengan el mismo formato para evitar así información errónea debido a que la mayoría de datos proviene de diferentes fuentes y, al unirlos, ésta debe estar estandarizada.
- No volátil: en los sistemas multidimensionales, la base de datos se optimiza para realizar consultas complejas, facilitando así el análisis de los datos y, en consecuencia, la toma de decisiones.
- De tiempo variante: los datos almacenados permiten obtener información precisa en determinados intervalos de tiempo.

El modelo dimensional conlleva una técnica de modelado que facilita la compresión de la base de datos, haciéndola intuitiva para usuarios no expertos. Se utiliza comúnmente para representar datos analíticos porque estructura los datos de una manera comprensible para el usuario de negocio, además de tener un alto rendimiento en las consultas. Dentro del modelo de datos dimensional destacan dos conceptos clave:

- Hechos: son las métricas, normalmente valores cuantitativos (numéricos) susceptibles de ser agregados.
- Dimensiones: son los valores cualitativos que proporcionan descripciones a los hechos, aportando un contexto a los mismos.

La tabla de hechos registra medidas o métricas de un evento específico, evitando repetir de manera completa los atributos dimensionales (la clave primaria está compuesta por los claves principales de las tablas de dimensiones) y se diseña según el nivel de granularidad deseado mientras que las tablas de dimensiones tienen una clave primaria simple, generalmente con un número bajo de registros aunque cada registro puede contener un gran número de atributos.

Existen dos técnicas para llevar a cabo el modelado dimensional: el esquema en estrella y el esquema en copo de nieve. El esquema en estrella es un modelo de datos formado por una tabla de hechos que contiene los datos para el análisis, rodeada de las tablas de dimensiones. Su principal ventaja es la simplicidad de las consultas que permite agregaciones más rápidas y un mejor rendimiento. Por otro lado, el esquema en copo de nieve es una estructura más compleja donde alguna de las dimensiones se implementa con más de una tabla de datos. Su principal ventaja es la reducción del espacio de almacenamiento al normalizar éstas (se elimina la redundancia). Sin embargo, esta normalización aumenta la complejidad de las consultas y tiene un peor rendimiento.

OLAP es una tecnología que nos permite organizar la información en una estructura dimensional o cubo, compuesto por dimensiones y medidas. Las dimensiones se utilizan para segmentar y dividir los datos mientras que las medidas proporcionan valores numéricos agregados. Proporciona una visión multidimensional de los datos, donde cada eje representa diferentes aspectos a analizar dentro de una organización o empresa.

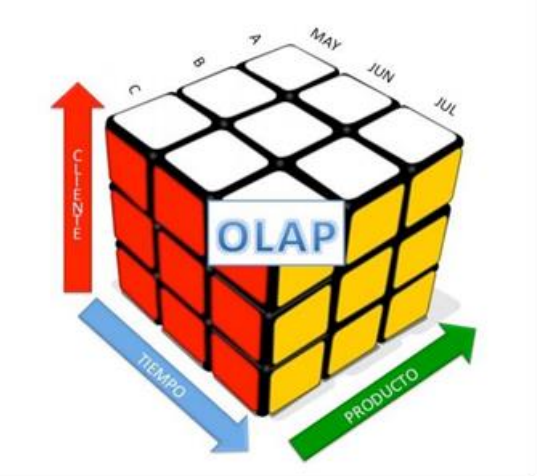

Figura 22. Cubo OLAP.

En los orígenes de la tecnología OLAP, la mayor parte de las compañías asumió que la única solución para una aplicación OLAP era un modelo de almacenamiento no relacional. Sin

embargo, otras compañías no tardaron en descubrir que se podrían utilizar también sistemas de administración de bases de datos relacionales. A la tecnología OLAP multidimensional se la denominó MOLAP mientras que a la tecnología OLAP relacional se la denominó ROLAP.

Las implementaciones MOLAP normalmente tienen mejor rendimiento y mayor velocidad que las ROLAP aunque tienen problemas de escalabilidad. Las implementaciones ROLAP son normalmente más atractivas para los clientes ya que, aparte de su mayor escalabilidad, aprovechan las inversiones ya realizadas en tecnologías de bases de datos relacionales.

Dependiendo de la forma de almacenamiento, los cubos OLAP se clasifican en:

- MOLAP: los datos se almacenan junto con sus agregaciones en una estructura multidimensional de alto rendimiento. Tiene un excelente rendimiento y comprensión de los datos, así como el mejor tiempo de respuesta. Muy apropiado para cubos de uso frecuente.
- ROLAP: toda la información del cubo, datos y agregaciones, se almacena en una base de datos relacional. Generalmente más lenta que MOLAP. Se usa normalmente con grandes conjuntos de datos que no son frecuentemente consultados.
- HOLAP: combina atributos de MOLAP y ROLAP, donde la agregación de los datos se almacena en una estructura multidimensional usada por MOLAP y los datos se almacenan en una base de datos relacional. Los cubos HOLAP son más pequeños que los MOLAP y responden más rápido que los ROLAP. Se utiliza generalmente para cubos que requieran una rápida respuesta y una gran cantidad de datos.

En la fase de explotación y visualización de los datos, la información almacenada se presenta de manera que pueda ser entendida y analizada para ayudar en la toma de decisiones. Permite hacer comparaciones cronológicas explotando una base de datos historificada. Esta comparación se podrá personalizar a través de distintos filtros, pudiendo representar un gran volumen de información en un simple cálculo, gráfica o tabla así como visualizar la información en diferentes niveles de agregación o jerarquías.

Dentro de las aplicaciones de BI de software libre (sin licencia de pago) disponibles en el mercado, Pentaho Community es la más extendida y utilizada. Es un conjunto de herramientas que permiten desarrollar todas las fases de un proyecto de BI, algunas de las cuales se describen brevemente a continuación:

- Data Integration: desarrollo y ejecución de procesos de ETL mediante un entorno gráfico denominado Spoon. También integrados en esta herramienta, Pan y Kitchen permiten también ejecutar dichos procesos mediante línea de comandos.
- Mondrian: servidor OLAP que gestiona la comunicación entre una aplicación OLAP y el almacén de datos. No tiene almacenamiento propio sino que trabaja con los datos almacenados en una base de datos relacional. Su función es traducir las consultas de algún lenguaje dimensional como, por ejemplo, MDX al lenguaje SQL y lanzarlas contra la base de datos relacional. También posee un manejo inteligente de la caché que

permite agilizar las consultas. En definitiva, un motor HOLAP que combina la flexibilidad de los motores ROLAP con una caché que le proporciona velocidad.

- Schema Workbench: creación de esquemas Mondrian de cubos OLAP mediante un entorno gráfico a partir de las tablas de hechos y dimensiones de un modelo dimensional.
- Saiku Analytics: visor OLAP para realizar análisis de modelos dimensionales (cubos OLAP) de forma fácil e intuitiva, permitiendo construir vistas arrastrando y soltando campos. Los datos se pueden analizar en forma de tabla pivotante o en modo gráfico.

Sin embargo, cuando el conjunto de datos presenta características de Big Data (volumen, variedad y velocidad), los requisitos de procesamiento y almacenamiento son muy exigentes incluso para tecnologías específicamente diseñadas para tal fin. Para soportar aplicaciones OLAP sobre Big Data, en los últimos años han aparecido distintas tecnologías, siendo algunas de las más conocidas Kylin, Vertica, Druid, Google Big Query y Amazon RedShift.

Aun siendo distintas sus respectivas arquitecturas, todas ellas permiten realizar consultas analíticas sobre Big Data con unos tiempos de respuesta (latencia) que van desde milisegundos hasta unos pocos segundos. De entre todas ellas, Kylin y Vertica destacan por sus resultados en multitud de implementaciones de sistemas OLAP de Big Data.

Kylin es un motor distribuido de procesamiento analítico. Proporciona una interfaz SQL y soporta un análisis multidimensional sobre la plataforma Hadoop. A continuación se enumeran algunas de sus características principales:

- Consultas interactivas sobre modelos multidimensionales con latencias inferiores al segundo.
- Soporte para el lenguaje estándar de consulta SQL.
- Integración con herramientas de explotación y visualización de datos como, por ejemplo, Power BI, Tableau o Pentaho.
- Escalabilidad y disponibilidad.
- Soporte para almacenamiento tanto en HDFS como en Amazon S3.
- Interfaz web para la gestión de los modelos multidimensionales, las cargas y actualizaciones de los cubos y la configuración, así como una RESTFUL API para la realización de consultas y la obtención de resultados en formato JSON.
- La versión de pago añade algunas características mejoradas como, por ejemplo, un sistema de almacenamiento de mayor rendimiento e incluye soporte técnico.

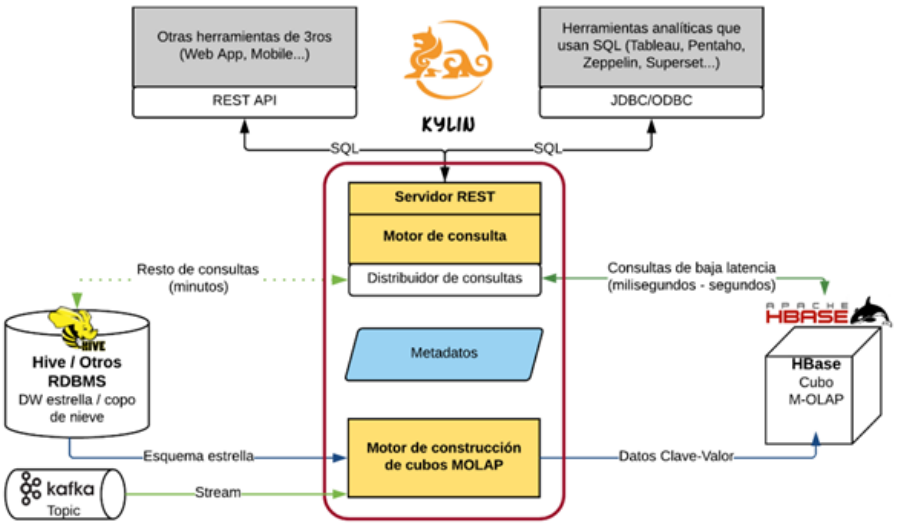

Figura 23. Arquitectura de Kylin.

Para el almacén de datos se puede utilizar tanto cualquier RDBMS como Hive o incluso Kafka. Una vez seleccionadas las fuentes de datos, hay que definir el modelo de datos dimensional a través de la interfaz web, indicando aspectos tales como las uniones entre las distintas tablas, las métricas y dimensiones a usar o los campos de fecha utilizados para la carga incremental y actualización de los datos. Posteriormente, se define el cubo OLAP, indicando aspectos tales como el nivel de preagregación y precombinación de los datos. Este tipo de arquitecturas OLAP basadas en la preagregación y precombinación de lo datos se denotan como MOLAP y logran excelentes tiempos de respuesta.

Vertica es un sistema de base de datos analítica con una arquitectura de procesamiento de datos masivos en paralelo (MPP), almacenamiento basado en columna. Está orientada a proporcionar funcionalidades OLAP sobre entornos de Big Data. A diferencia de Kylin, Vertica es una tecnología que no necesita de Hadoop para su funcionamiento. A continuación se enumeran algunas de sus características principales:

- Consultas interactivas sobre miles de millones de filas.
- Soporte para el lenguaje estándar de consulta SQL.
- Integración con herramientas de explotación y visualización de datos como, por ejemplo, Power BI, Tableau o Pentaho.
- Escalabilidad y disponibilidad.
- Interfaz web para la gestión y carga del clúster, las bases de datos y los esquemas.
- Funciones de aprendizaje automático.
- Capacidad de definir funciones por el usuario (UDF) para usarlas dentro de consultas SQL, empleando lenguajes tales como Java, C++, Python o R.
- La versión gratuita está limitada en capacidad de procesamiento a un clúster de 3 máquinas y en capacidad de almacenamiento a 1 TB.

Figura 24. Arquitectura de Vertica.

Para crear un cubo OLAP primeramente hay que definir un esquema de tablas y cargarlo desde las fuentes de datos soportadas como, por ejemplo, RDBMS, herramientas de ETL como Pentaho o Talend o incluso Kafka para la carga de datos en tiempo real. Posteriormente pueden ejecutarse las consultas SQL o conectarse a través de algunas de las herramientas de explotación y visualización de datos más populares como, por ejemplo, Power BI, Tableau o Pentaho para crear cuadros de mando e informes.

A diferencia de Kylin, Vertica no preagrega ni precombina los datos, por lo que la ejecución de las consultas analíticas requiere mayor computación en tiempo de consulta. Sin embargo, mediante un asistente de optimización se puede lograr cierto nivel de preagregación y precombinación, mejorando así los tiempos de respuesta. Este tipo de aproximaciones se denotan como HOLAP.

Además, el clúster de Vertica es capaz de integrarse con un clúster Hadoop para hacer consultas sobre datos almacenados en Hive o HDFS o aprovechar herramientas como Spark para el procesamiento.

A continuación se muestran sendas comparativas entre Kylin y Vertica sobre los tiempos de respuesta sobre distintos volúmenes de datos.

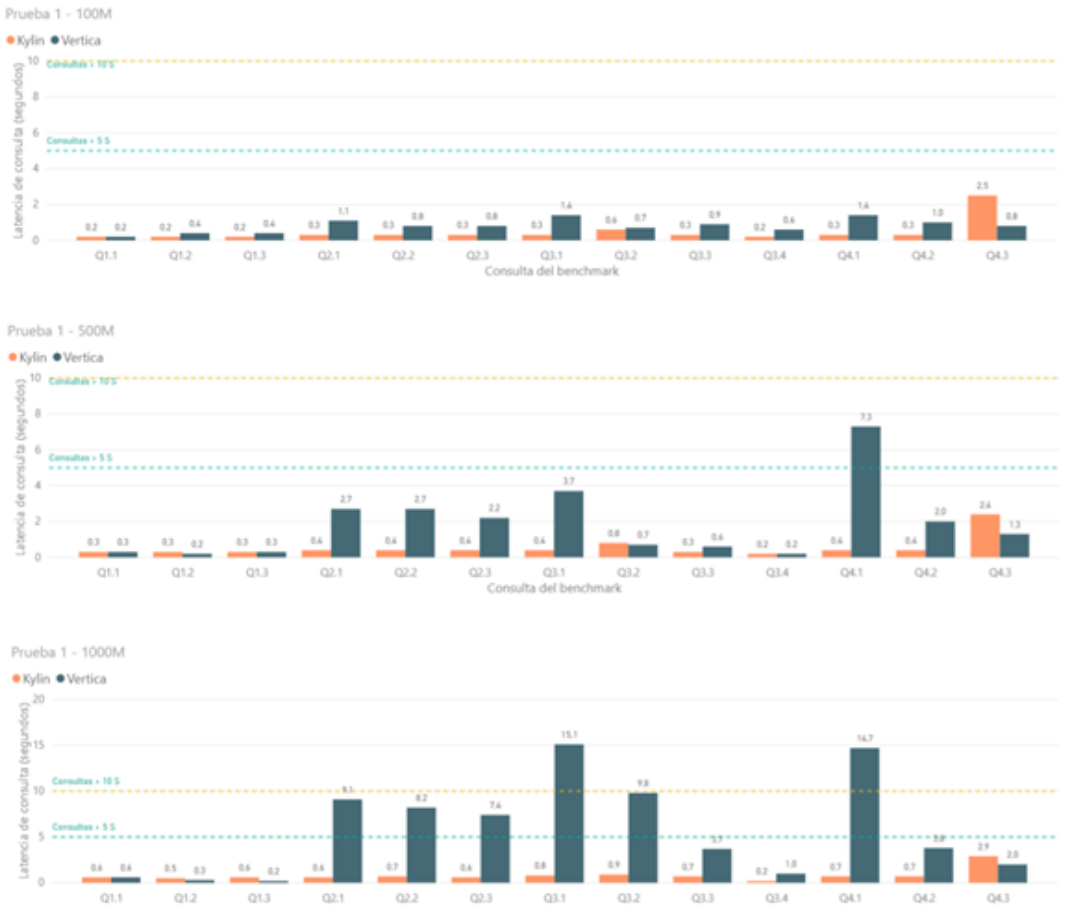

Figura 25. Comparativa entre Kylin y Vertica sobre los tiempos de respuesta sobre distintos volúmenes de datos (I).

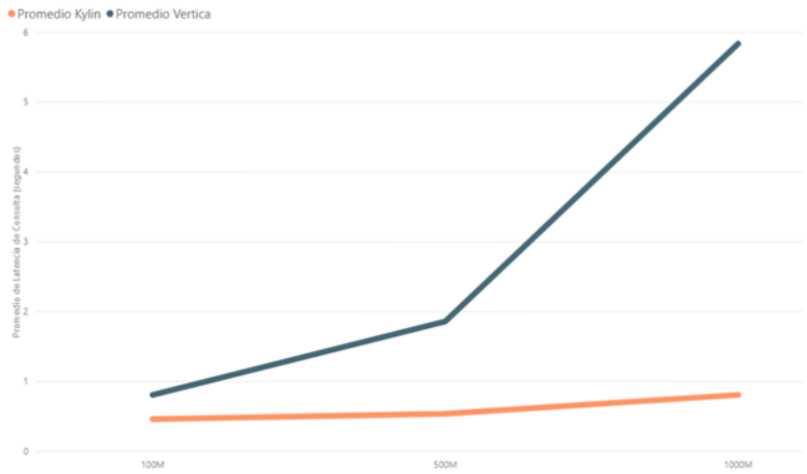

Figura 26. Comparativa entre Kylin y Vertica sobre los tiempos de respuesta sobre distintos volúmenes de datos (II).

Como se observa en la figura anterior, el tiempo de respuesta en el caso de de Kylin apenas se incrementa aun aumentando considerablemente el volumen de datos. Esto es debido a la preagregación y precombinación de los datos, de tal forma que el procesamiento y las operaciones de I/O necesarias en tiempo de consulta son mucho menores que en el caso de Vertica. Sin embargo, hay que tener en cuentas otros factores importantes además de la latencia. Uno de ellos es el tiempo de carga de los datos. En el caso de Kylin, la mayor parte del procesamiento se realiza en tiempo de carga (fase de construcción del cubo) para así preagregar y precombinar los datos. Para reducir este tiempo de carga, Kylin necesita disponer de un mayor número de recursos hardware en el clúster Hadoop.

1.4 Sistemas de aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial cuyo objetivo es desarrollar algoritmos capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos. Su estudio se centra en la complejidad computacional de los problemas. Gran parte de la investigación realizada en este ámbito está enfocada al diseño de soluciones factibles a problemas de tipo NP-complejo.

El aprendizaje automático tiene como resultado un modelo para resolver una tarea determinada. Los distintos modelos se pueden clasificar como:

- Geométricos: se construyen en el espacio de características, pudiendo tener múltiples dimensiones.
- Probabilísticos: determinan la distribución de probabilidad de la función que describe los valores de las variables dependientes con respecto de las independientes.
- Lógicos: transforman y expresan las probabilidades en reglas organizadas en forma de árboles de decisión.

Los modelos se pueden clasificar también como modelos de agrupamiento o gradiente. Los primeros tratan de dividir el espacio de características en diferentes grupos mientras que los segundos calculan el gradiente con respecto del cual se puede diferenciar las respectivas características.

Según el tipo de aprendizaje, los algoritmos se pueden clasificar como:

- Aprendizaje supervisado: obtiene un modelo a partir de la correspondencia entre las entradas y las salidas del sistema.
- Aprendizaje no supervisado: el modelado se lleva a cabo únicamente a partir de las entradas del sistema. No se tiene información de las respectivas salidas.
- Aprendizaje por refuerzo: el algoritmo aprende observando el mundo que le rodea. Las entradas del sistema son el feedback que obtiene como respuesta a sus acciones. Aprende a base de ensayo y error.

A continuación se describen brevemente algunos de los principales algoritmos de aprendizaje automático:

- Árboles de decisión: modelo predictivo basado bien en la impureza de Gini (algoritmo CART) o bien en la entropía de la información (algoritmos ID3 y C4.5) en cada nodo del árbol de decisión.
- Redes neuronales: paradigma de aprendizaje automático inspirado en las neuronas del sistema nervioso. Constituye un sistema de enlaces entre distintas neuronas que colaboran entre sí para producir un estímulo de salida donde las conexiones tienen pesos numéricos que se adaptan según la experiencia.
- Máquinas de vectores soporte: modelo predictivo discriminatorio que a partir de un conjunto de ejemplos de entrenamiento permite clasificar entre distintas categorías.
- Algoritmos de agrupamiento: método de aprendizaje no supervisado que clasifica las observaciones en distintos clústers según una determinada medida de similitud.

Algunas de las librerías, plataformas o lenguajes de programación más populares en el ámbito del aprendizaje automático se describen brevemente a continuación:

- Scikit-learn: Librería de aprendizaje automático de software libre para el lenguaje de programación Python que incluye una amplia variedad de algoritmos de clasificación, regresión y agrupamiento así como multitud de funciones de utilidad. Está implementada sobre las siguientes las librerías NumPy (librería numérica para operar con vectores multidimensionales), Pandas (librería de manipulación y análisis de los datos), SciPy (librería científica), Matplotlib (librería de visualización), Ipython (consola interactiva) y SymPy (librería numérica simbólica).

Figura 27. Scikit – learn.

- R: lenguaje de programación interpretado orientado al análisis estadístico disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux. Proporciona un amplio abanico de herramientas estadísticas y gráficas. Soporta distintos tipos de paradigmas de programación. En sí mismo es un lenguaje imperativo aunque soporta también tanto programación orientada a objetos como funcional. Al ser un lenguaje de código abierto cuenta con una extensa comunidad activa de colaboradores que desarrollan nuevas librerías extendiendo su funcionalidad.
- H2O: plataforma de software libre desarrollada para combinar aprendizaje automático y Big Data, siendo la escalabilidad su principal característica. Sigue el paradigma clave-valor para el almacenamiento de los datos y MapReduce para su procesamiento. Mediante distintas APIs se puede acceder desde R, Python o Scala, así como a través de una interfaz web similar al Jupyter notebook denominada Flow. Puede interactuar

con HDFS, Amazon S3 o distintos sistemas de base de datos NoSQL. Una de sus herramientas, Sparkling Water, combina el motor de procesamiento de datos Spark con la propia plataforma de H2O.

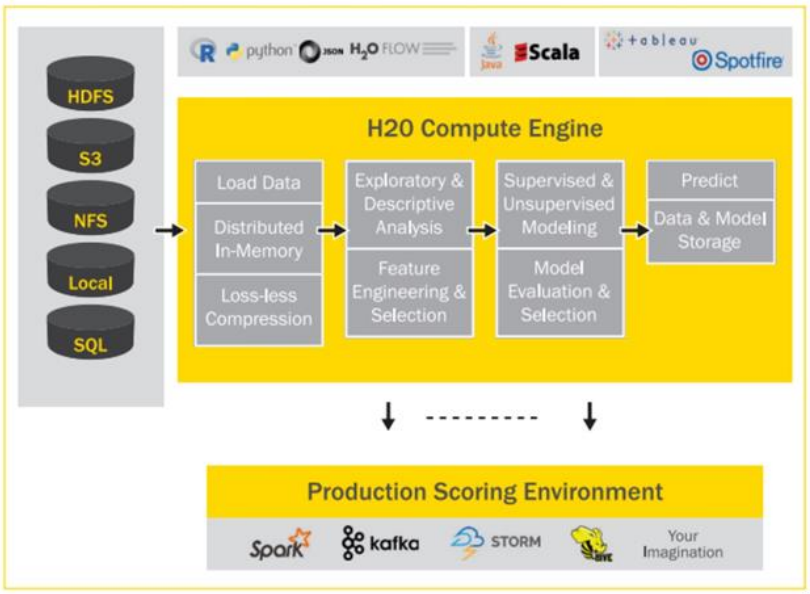

Figura 28. Arquitectura de H2O.

- Mahout: librería de software libre con implementaciones escalables de una amplia variedad de algoritmos de aprendizaje automático basada en el paradigma de procesamiento MapReduce del ecosistema Hadoop. Estos algoritmos están orientados fundamentalmente a tareas de clasifiación, agrupamiento y recomendación.

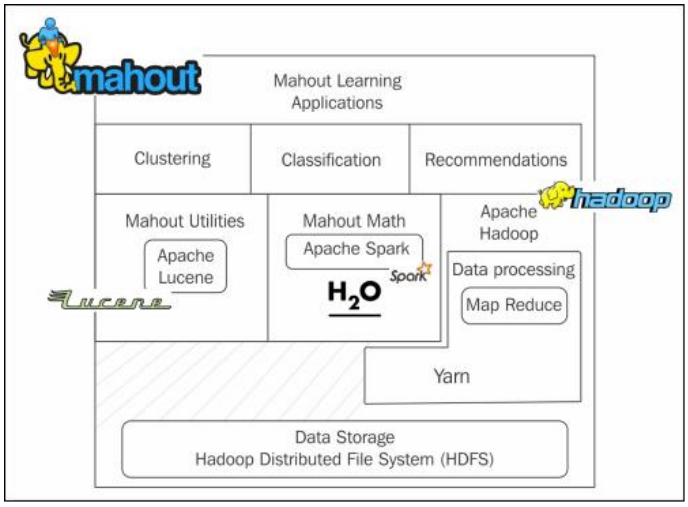

Figura 29. Arquitectura de Mahout.

- MLlib: API de Spark implementada sobre RDD, interoperable tanto con librerías de Python como de R. Incluye una amplia variedad de algoritmos de clasificación, regresión, agrupamiento y de recomendación así como multitud de funciones de

utilidad. Los datos pueden estar almacenados en RDBMS o sistemas de base de datos NoSQL así como en el ecosistema Hadoop como, por ejemplo HDFS o HBase.

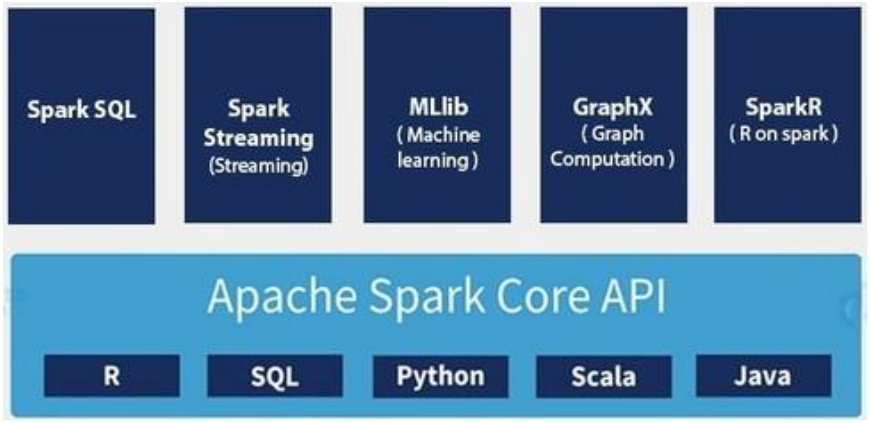

Figura 30. Arquitectura de Spark.

- TensorFlow: librería de código abierto para aprendizaje automático basado en redes neuronales de aprendizaje profundo. Disponible para Linux y macOS así como para Android e iOS. Se puede ejecutar sobre múltiples CPUs y GPUs. Utiliza gráficos de flujo de datos donde los nodos representan operaciones matemáticas y las aristas, vectores de datos multidimensionales (tensores). Soporta una amplia variedad de lenguajes de programación como, por ejemplo, Java, Scala y Python.

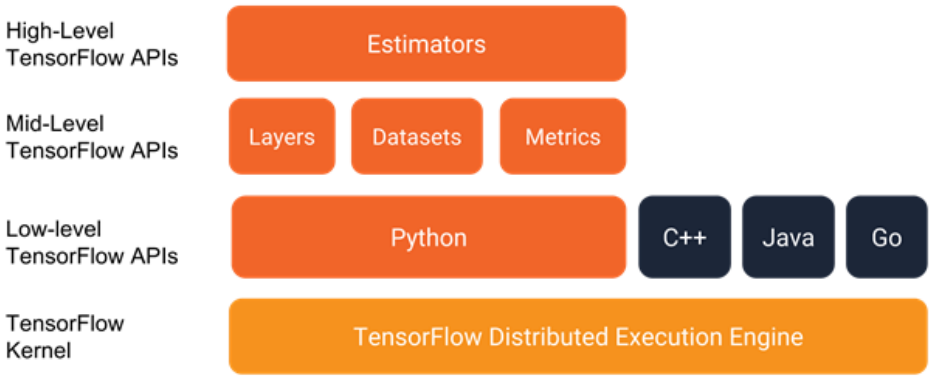

Figura 31. Arquitectura de TensorFlow.

- Keras: librería de código abierto para aprendizaje automático basado en redes neuronales de aprendizaje profundo escrita en Python. Está concebida para actuar como una interfaz en lugar de ser una plataforma de aprendizaje automático por sí sola. Ofrece un conjunto de abstracciones, más intuitivas y de alto nivel, que hacen más sencillo el desarrollo de modelos de aprendizaje profundo, independientemente del backend utilizado como, por ejemplo, TensorFlow.

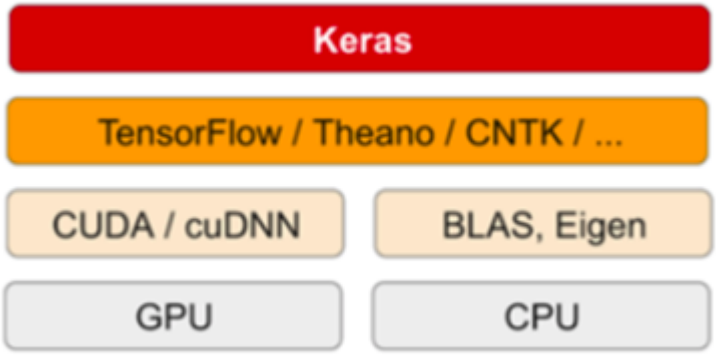

Figura 32. Arquitectura de Keras.

- PyTorch: librería de código abierto para aprendizaje automático basado en redes neuronales de aprendizaje profundo escrita en Python. Dispone de una interfaz muy sencilla para la creación de redes neuronales. Permite operar de forma directa con tensores sin la necesidad de una librería de alto nivel como pueda ser Keras para TensorFlow. Dispone de soporte para su ejecución en tarjetas gráficas (GPU) utilizando internamente CUDA, una API que conecta la CPU con la GPU desarrollada por NVIDIA. A diferencia de TensorFlow, los grafos de computación son dinámicos en lugar de estáticos, pudiendo modificarlos en tiempo de ejecución. Esto facilita la depuración del código y hace más ágil su implementación.