



IFCT128PO: Big data

Unidad 3: Algunos conceptos técnicos de la analítica tradicional

ÍNDICE

INTRODUCCIÓN.....	3
OBJETIVOS / CAPACIDADES.....	4
1. EL TEOREMA DE BREWER.....	5
2. LAS NUEVAS BASES DE DATOS.....	7
3. PROCESAMIENTOS DISTRIBUIDOS. MAPREDUCE.....	10
2.1. Tipos de Bases de Datos NoSQL	7
3.1. Funcionamiento de MapReduce	10
3.2. ¿Qué elementos son clave para la puesta en marcha de MapReduce?.....	11
4. HERRAMIENTAS PARA FINES OPERACIONALES VS. ANALÍTICOS	12
RESUMEN.....	16
MAPA CONCEPTUAL.....	17
GLOSARIO	18

A lo largo de la unidad 3 haremos un recorrido de carácter más técnico por el ámbito de la **analítica de datos**. Para ello, es lógico no solo abordar las virtudes del panorama, sino también conocer las dificultades que nos plantea esta metodología.



En este sentido, comenzaremos conociendo el **Teorema de Brewer** para conseguir partir de la base de unas dificultades previas de cara al análisis y continuar con las diferentes bases de datos que podemos encontrarnos para empezar a trabajar en nuestro proyecto de **big data**.

Tras este recorrido, abordaremos el procesamiento distribuido de datos de la mano de **MapReduce** y qué diferencias existen entre un proyecto de big data con fines operacionales o analíticos. Por supuesto, no tiene el mismo uso uno y otro.

En esta unidad te ayudaremos a distinguir.



En esta unidad de aprendizaje, las capacidades que más se va a trabajar son:

- ✓ Conocer la problemática que encuentra el big data a la hora de realizar el almacenamiento masivo, recogida en el Teorema de Brewer o teorema CAP.
- ✓ Adquirir conocimientos sobre los diferentes tipos de bases de datos disponibles en el mercado.
- ✓ Saber qué funciones realiza MapReduce.
- ✓ Saber diferenciar entre big data para fines analíticos u operacionales.

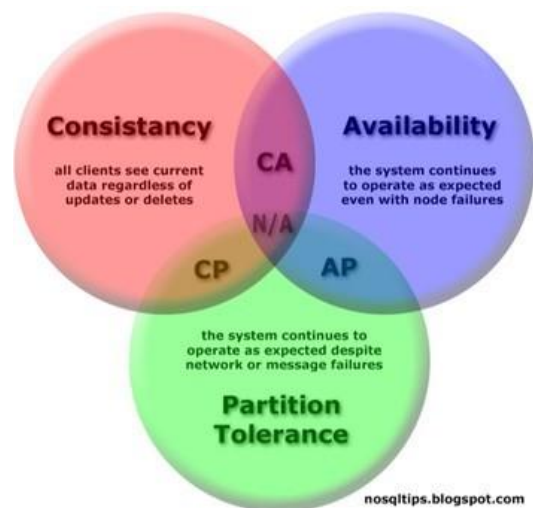




A nivel tecnológico, la **problemática del big data** encuentra su solución gracias al almacenamiento masivo -para así saber el volumen de crecimiento que podemos alcanzar- y el procesamiento distribuido. Pero esta solución también tiene un problema. Una dificultad que se explica en el **Teorema de Brewer o teorema de CAP**. Este teorema dice que en sistemas distribuidos se da una imposibilidad clara de dar cabida a la vez a la consistencia, disponibilidad y tolerancia a particiones.

Pero ¿a qué hacen referencia estos conceptos?:

- **Consistencia:** al llevar a cabo una consulta siempre tiene que obtenerse la misma información, más allá del nodo o servidor que realice la petición en cuestión.
- **Disponibilidad:** que todos los datos estén a disposición de los usuarios aunque uno de los nodos haya fallado.
- **Particiones:** los sistemas distribuidos pueden presentar divisiones en particiones, por lo que influye en que el sistema tiene que seguir en funcionamiento aunque aparezcan pequeñas caídas que dividan el sistema.
- **Marcas de categorías:** esta problemática para cumplir estas tres características da lugar a que existan bases de datos que resolverán mejor unas áreas que otras. A continuación detallamos diferentes marcas que funcionan mejor para cada categoría:
 - ✓ **1. Consistencia:** BigTable, HyperTable, HBase, MongoDB, Redis, MemCacheDB, Scalaris.
 - ✓ **2. Disponibilidad:** RDBMS, GreenPlum.



- ✓ **3. Tolerancia a particiones:** Dynamo, CouchDB, Cassandra, SimpleDB, TokyoCabinet, Riak, Voldemort.

2. LAS NUEVAS BASES DE DATOS



Antes del surgimiento del big data solo existían bases de datos relacionadas con entornos SQL. Es decir, bases de datos con lenguaje de consulta estructurado. Estas bases se rigen por una serie de parámetros denominados ACID: Atomicity, Consistency, Isolation y Durability. Sin embargo, como ya sabemos, el big data no se vale exclusivamente de datos estructurados y de ser así no tendría sentido. Por lo tanto, han ido apareciendo nuevas bases de datos no relacionales -conocidas como NoSQL- que dan solución a la problemática del rendimiento y la escalabilidad.

A continuación podemos cuáles son las **características comunes** de cualquier base de dato NoSQL.

- ❓ **Escalabilidad horizontal:** se pueden añadir más nodos para aumentar su rendimiento y se puede señalar los que están activos.

- ❓ No necesitan mucha computación por lo que **el coste se reduce**.

- ❓ **Dan tratamiento a un gran volumen de datos:** su estructura es distribuida.

- ❓ **Estos sistemas utilizan lo que se denomina BASE:** Basically Available, Soft state, and Eventual Consistency.



2.1. Tipos de Bases de Datos NoSQL

- **Bases de datos clave-valor:** Es el tipo más popular de bases de datos NoSQL. Se caracteriza por su sencillez y porque cada elemento está identificado por una única clave. **{clave:valor}**

✓ **Ventajas:** hace muy fácil la recuperación de la información y muestra eficiencia en la lectura y la escritura.

→ **Bases de datos documentales:** Se trata de las bases de datos con más versatilidad. Utiliza normalmente una estructura como JSON o XML y trabaja con una clave única por registro. Este tipo de base de datos puede llevar a cabo consultas por clave-valor o relacionadas con el contenido del documento aportado.

Son muy útiles pero hay que efectuar varios reports y éstos tienen que unirse de manera dinámica porque sus elementos van cambiando constantemente.

→ **Bases de datos en grafo:** Son bases de datos que por definición se encargan de encontrar nexos de unión entre los diferentes datos para extraer el auténtico valor de los mismos. Los datos se representan como nodos de un grafo.

Estas bases de datos son idóneas si los datos están fuertemente relacionados como por ejemplo, en redes sociales, en recomendaciones en tiempo real, para detectar fraudes, etc.

→ **Bases de datos orientados a columnas:** La diferencia con las bases de datos relacionales es que los datos son guardados en diferentes filas. Se utilizan para entornos en los que se necesite acceder a diferentes columnas con muchas filas y que tengan poca escritura.

Son muy funcionales, por ejemplo, para análisis de datos y en el procesamiento de eventos y su análisis. También para aplicaciones que necesitan una distribución geográfica en varios centros de datos, aplicaciones que tengan campos dinámicos o que puedan soportar cierta inconsistencia en las réplicas pero en poco tiempo; o aplicaciones con un volumen de datos muy elevado.

En la siguiente tabla podemos ver un pequeño resumen de las principales características de las principales bases de datos NoSQL.

Tipo	Rendimiento	Escalabilidad	Complejidad	Flexibilidad
Clave-valor	Elevado	Elevada	Baja	Elevada
Columna	Elevado	Elevada	Baja	Moderada
Documentales	Elevado	Variable	Baja	Elevada
<u>En</u> grafo	Variable	Variable	Elevada	Elevada

→ **Modelo híbrido:** También existen soluciones híbridas que acogen tanto modelos relacionales como los no relacionales. Entre ellas, CortexDB, Foundation DB y Orient DB ofrecen diferentes modelos NoSQL.

3. PROCESAMIENTOS DISTRIBUIDOS. MAPREDUCE



Para adentrarnos en el mundo de **MapReduce** es necesario acercarnos antes a la definición del concepto de procesamiento distribuido. Hablamos de procesamiento distribuido para referirnos a una manera de proceso en la que se distribuyen los datos y funciones en diferentes elementos de un sistema. Esto conlleva la utilización de una red o área local o, bien, una red de tarea amplia.

En este sentido, uno de los principales retos con los que se encuentra el big data es la realización de una distribución de aplicaciones vinculada a un gran volumen de datos. Dentro de esta problemática nace MapReduce, una tecnología que ofrece a los desarrolladores la posibilidad de **crear programas que**

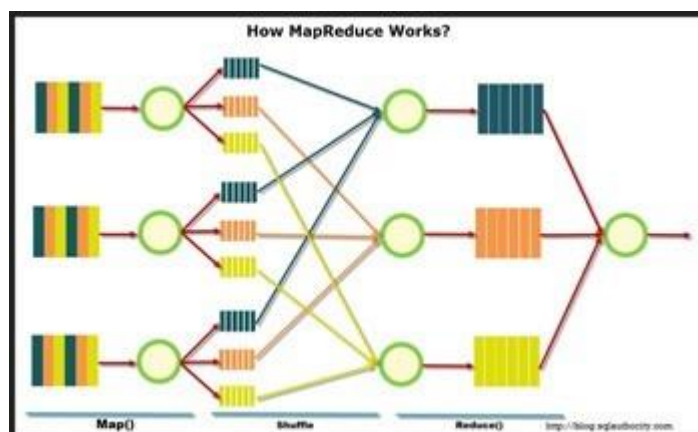


tengan la capacidad de procesar un alto volumen de datos no estructurados a la vez, gracias a un conjunto distribuido de procesadores.

3.1. Funcionamiento de MapReduce

Map y Reduce son **dos funciones** que ya existían previamente en otros programas. Sin embargo, en este caso la clave es la combinación de ambas.

→ **Función Map**: los datos son procesados uno a uno para



transformarse en un data set o conjunto de datos independiente.

→ **Función Reduce**: convierte los datos que han sido preparados por la función Map generando un nuevo valor con los resultados.

De manera sencilla, podemos decir que **MapReduce consiste** en:

1. Dividir el cómputo de todos los datos en partes más pequeñas.
2. Hacer el procesamiento de estos datos de manera paralela en varias máquinas.
3. Añadir los resultados para luego devolverlos.

3.2. ¿Qué elementos son clave para la puesta en marcha de MapReduce?

A continuación, se exponen los **elementos claves** para la puesta en marcha de MapReduce. No obstante, ninguno de estos elementos son útiles si no se utiliza la tecnología apropiada:

- **Planificación** o Scheduling: MapReduce jobs obligatoriamente tiene que partirse en tareas individuales para ejecutar la función Map y Reduce. Será fundamental tener en cuenta los tiempos para verificar que todas las tareas mapping estén acabadas antes de empezar con la función reduce.
- **Sincronización**: comprobación de que todos los procesos paralelos se están llevando a cabo correctamente para así comenzar el reducing. Los datos intermedios quedarán copiados en la red y se produce el conocido "shuffle and sort".

- **Código-Data:** en este punto las funciones de mapping ya pasan a formar parte de la misma máquina en la que ya están guardados los datos que van a ser procesados.
- **Gestión de errores:** un gran porcentaje de computadoras MapReduce reconocen los errores producidos durante el proceso y llevan a cabo la corrección que necesitan.



*Una de las herramientas más conocidas es **Hadoop** (herramienta libre), que se ha proclamado como uno de los bloques más importantes para la captación y procesamiento en big data. Su función es agilizar el proceso poniendo en paralelo en diferentes máquinas los procesos de datos que se estén llevando a cabo en ese mismo momento.*

Dentro de esta tecnología existen **dos componentes**:

- ❑ **Hadoop Distributed File System:** Fiable y de bajo coste, este sistema de almacenamiento simplifica la gestión de archivos en varias computadoras.
- ❑ **MapReduce engine:** Implementa MapReduce a través de un proceso de rendimiento elevado desarrollado de manera paralela.

4. HERRAMIENTAS PARA FINES OPERACIONALES VS. ANALÍTICOS

Tras los puntos anteriores podemos afirmar que la tecnología escogida dependerá en primer lugar, siempre, de las necesidades que tenga la empresa o entidad que desarrolle el **proyecto**. *Es decir, si hay necesidad de llevar a cabo operaciones en tiempo real, con herramientas que tengan en*




consideración si los datos con los que cuenta son guardados o capturados o si el sistema de análisis de los datos históricos va a conllevar algún tipo de complejidad.

En referencia a todo ello, debemos clasificar las necesidades de la empresa en dos para desarrollar nuestro proyecto. Así, diferenciaremos si se trata de **big data analítico** o **big data operacional**.

Según se trate de uno u otro tipo de empleo de big data, la **arquitectura tecnológica** se verá modificada en varios aspectos. No obstante, en ambos casos se suele operar sobre una amplia cantidad de servidores en un conjunto de ordenadores unidos entre sí, es decir, un clúster, para poder canalizar un gran volumen de datos a través de millones de registros.

En el siguiente cuadro, podemos ver las diferencias principales entre ambos tipos de uso de big data:

→ **Big data operacional:**

<p>Big Data Operacional</p> <p>Los sistemas que se aplican a Big Data operacional se centran en las solicitudes simultáneas a la vez que presentan una baja latencia de respuestas relacionadas con aspectos de acceso, pero muy selectivos.</p> 	Latencia	1 ms-100 ms
	Concurrencia	1000-100,000
	Queries	Selectiva
	Acceso	Lee y escribe
	Tipos de datos	Operacionales
	Tecnología	NoSQL
	Usuario final	Cliente


Ejemplo:

Un ejemplo práctico de big data operacional es el uso que le está dando el sector de la banca a esta tecnologías. Hoy en día, los bancos cuentan con un importante volumen de datos de sus clientes a través de sus compras, cuándo las realizan, desde dónde, qué tipos de gastos tienen diariamente o cómo se van recalculando sus hipotecas de tipo variable, por ejemplo. Gracias a estos datos recogidos en tiempo real, los bancos están personalizando cada uno de los servicios que



ofrecen a través de cada una de sus comunicaciones con el cliente.

→ **Big data analítico:**

<p>Big Data Analítico</p> <p>Se caracterizan por basarse en el alto rendimiento realizando consultas de gran complejidad y permitiendo tocar un alto porcentaje de datos del sistema en el momento en el que se necesite.</p> 	Latencia	1 min-100 min
	Concurrencia	1-10
	Queries	No selectiva
	Acceso	Escribe
	Tipos de datos	Históricos
	Tecnología	MapReduce, MPP Database
	Usuario final	Científico de datos

Ejemplo:



En este caso, un buen ejemplo práctico del uso de big data en este caso sería el análisis que se realiza de la satisfacción del cliente a través de las opiniones que dejan en las encuestas lanzadas por las empresas, los mensajes que dejan en redes sociales, etc. A través de la analítica de estos datos históricos, la empresa puede realizar un análisis de percepciones para determinar el sentimiento del usuario.



En **Recursos para ampliar** podrás encontrar más información sobre el Big Data Analytics.

Actividad de aprendizaje 1.



ACTIVIDAD

En una empresa familiar que lleva 20 años en el negocio del aceite, se llega a la conclusión de que sería conveniente implementar un proyecto de big data para conseguir mejores resultados a través de la optimización en la producción de su producto. Debate con tus compañeros qué tipo de big data debe emplear la empresa, si operacional o analítico. En base a tu respuesta, ¿cómo será la arquitectura tecnológica del mismo?

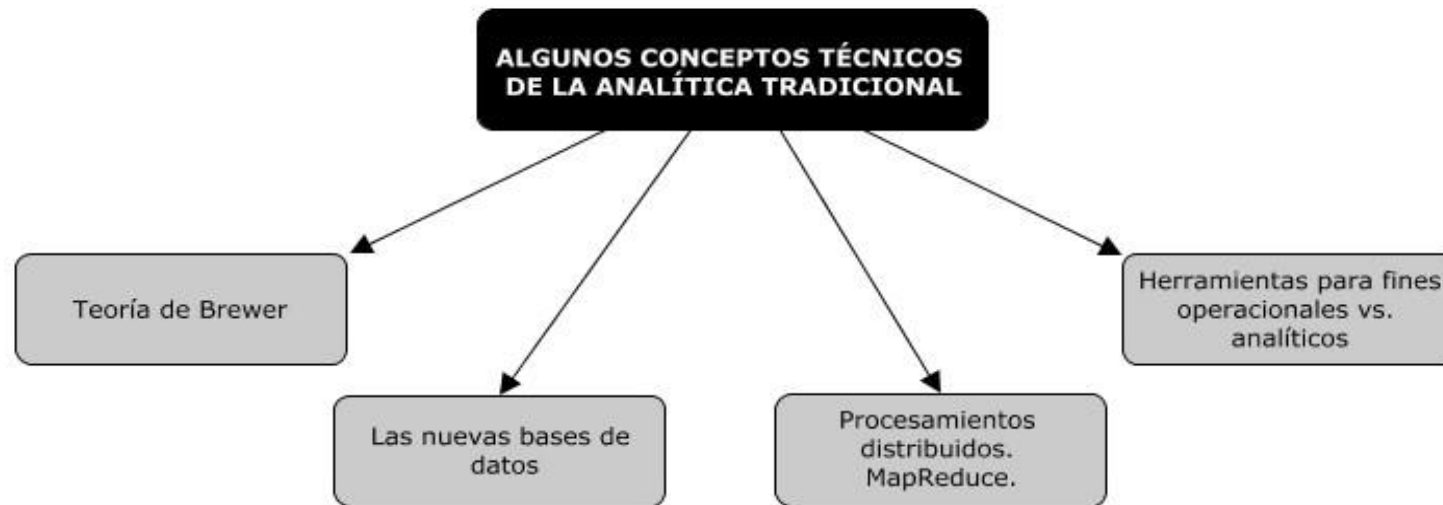
RESUMEN


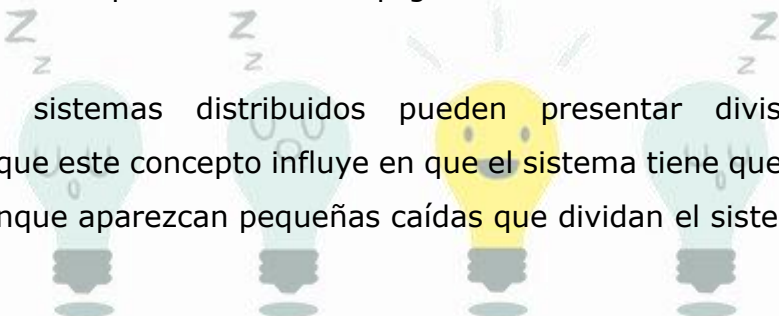
Como en toda metodología, en **big data** nos encontramos ciertos problemas que deberemos tener en cuenta de cara al planteamiento de nuestro proyecto. Como hemos visto, antes de empezar nuestro proyecto y con el fin de conseguir nuestros objetivos deberemos plantearnos si nos encontramos ante un caso propio para **perseguir fines operacionales** o si por el contrario, es más productivo **atenernos a fines puramente analíticos**.

A partir de este momento, tendremos que centrar nuestros esfuerzos en saber estructurar **qué elementos y qué tipo de procesamiento de datos** necesitamos y cómo para, de esta manera, 'centrar el tiro' y enfocarnos al 100% en encontrar los resultados más satisfactorios a nuestras necesidades para la toma de decisiones posterior, que cambie el rumbo de nuestra estrategia.

MAPA CONCEPTUAL

Para concluir la unidad la podemos resumir los **apartados e ideas principales** por medio de este mapa conceptual:



- 
- ❓ **Bases de datos NoSQL:** Bases de datos no relacionales que se utilizan para datos no estructurados. Se caracterizan por su amplio desarrollo, su nivel bajo de latencia, su escalabilidad horizontal, su estructura distribuida. Se rige por los parámetros BASE: Basically Available, Soft state, and Eventual Consistency.
- ❓ **Bases de datos SQL:** Bases de datos relacionales que nos permiten realizar diferentes clases de operaciones. Trabajan con un lenguaje de consulta estructurado basado en una serie de características denominadas ACID: Atomicity, Consistency, Isolation y Durability.
- ❓ **Big data analítico:** Se caracterizan por basarse en el alto rendimiento realizando consultas de gran complejidad y permitiendo tocar un alto porcentaje de datos del sistema en el momento en el que se necesite.
- ❓ **Big data operacional:** Los sistemas que se aplican a big data operacional se centran en las solicitudes simultáneas a la vez que presentan una baja latencia de respuestas relacionadas con aspectos de acceso pero muy selectivos.
- ❓ **Consistencia:** Al llevar a cabo una consulta siempre tiene que obtenerse la misma información, más allá del nodo o servidor que realice la petición en cuestión.
- ❓ **Disponibilidad:** Significa que todos los datos estén a disposición de los usuarios aunque uno de los nodos haya fallado.
- ❓ **Función Map:** Función dentro del procesamiento de datos gracias a la cual estos son procesados uno a uno para transformarse en un data set independiente.
- ❓ **Función Reduce:** Función dentro del procesamiento de datos que convierte los que han sido preparados por la función Map generando un nuevo valor con los resultados.
- ❓ **Particiones:** Los sistemas distribuidos pueden presentar divisiones en particiones. Por lo que este concepto influye en que el sistema tiene que seguir en funcionamiento aunque aparezcan pequeñas caídas que dividan el sistema.
- 

❓ **Teorema de Brewer o teorema de CAP:** Teorema que proclama que en sistemas distribuidos se da una imposibilidad clara de dar cabida a la vez a la consistencia, disponibilidad y tolerancia a particiones.

