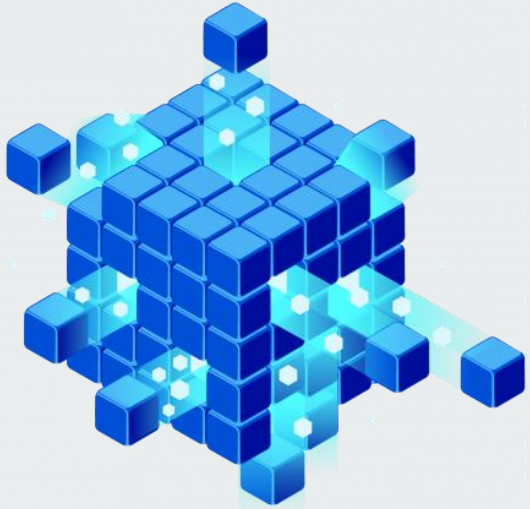


Sistemas de Big Data

UT01 Introducción al Big Data

Juan F. García Hinojosa

UT01 Introducción a Big Data

1. Por qué Big Data
    1. Las 5 Vs
    2. Qué conseguimos gracias a Big Data
2. Clusters de computadoras
3. Conceptos de almacenamiento de datos
    1. Base de Datos Relacional
    2. Dataset
    3. Almacén de Datos
    4. ACID
    5. Teorema CAP
    6. BASE
4. Conceptos de procesamiento de datos
    1. Procesamiento en paralelo
    2. Procesamiento distribuido
    3. Estrategias de procesamiento de datos
    4. OLTP
    5. OLAP
    6. Principio SCV
5. La arquitectura por capas de Big Data
6. El paisaje de Big Data

CASO PRÁCTICO

1 ¿Por qué Big Data?

- FL Logistics, la empresa que Felisa y Luís crearon hace ya 15 años, no para de crecer.
- Acaban de abrir su décimo almacén en España, siguen contratando nuevo personal y cada vez reciben más trabajo.
- Sin embargo, precísamente ese crecimiento que para ellos siempre fue un sueño a alcanzar se está convirtiendo en una gran preocupación. Aunque el cometido de su compañía pueda verse como el mover mercancías de un lugar de origen a otro de destino, el funcionamiento de la misma depende en gran medida de ser capaces de capturar, almacenar, gestionar y a ser posible también analizar una gran cantidad de datos (de proveedores, productos, orígenes, destinos, rutas, empleados, ...).
- Para el primer almacén que abrieron compraron un servidor de los más potentes del mercado, con el cuál dieron también soporte en un principio al segundo almacén.
- Al abrir el tercer almacén vieron que el servidor no era capaz de trabajar con tal carga de información, por lo que optaron por poner un servidor distinto en cada almacén. Eso complicó las cosas porque además necesitaron otro servidor extra en el que poder centralizar trabajo y los resultados de los análisis realizados por los servidores de los diversos almacenes.
- Ahora que están abriendo el décimo almacén, el servidor central ya está saturado, lo cual parece imponer un límite al crecimiento de la empresa.

¿Cómo van a poder tratar con tal cantidad de información si las cosas siguen yendo bien y necesitan seguir abriendo más almacenes?

1 ¿Por qué Big Data?

- Las metodologías y tecnologías para Big Data (macrodatos) aparecen como **respuesta** a la necesidad de **tratar cantidades de datos** tan **grandes** que desbordan los sistemas convencionales monomáquina.
- El uso moderno del término "big data" tiende a referirse al **análisis del comportamiento** del usuario, extrayendo valor **de los datos almacenados**, y formulando **predicciones** a través de los patrones observados.

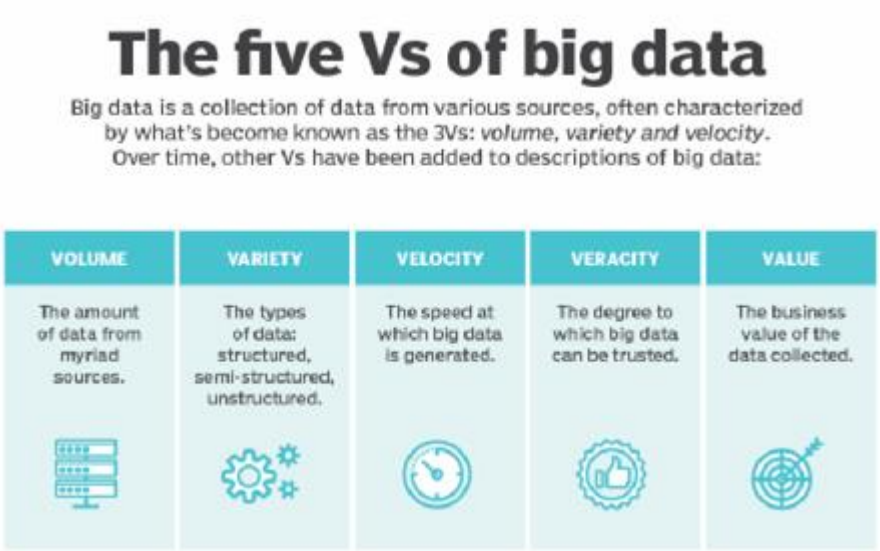
- Esta disciplina se ocupa de todas las actividades relacionadas con los sistemas que manipulan grandes conjuntos de datos.
- El límite superior de procesamiento ha ido creciendo a lo largo de los años. Se estima que el mundo almacenó unos 5 zettabytes en 2014. Si se pone esta información en libros, convirtiendo las imágenes y todo eso a su equivalente en letras, se podría hacer 4500 pilas de libros que lleguen hasta el sol.

1.1 Las 5 Vs

En esta sección hablaremos de **las cinco características de Big Data** que suelen emplearse para indicar si el procesamiento de datos que necesitamos realizar puede realmente considerarse como Big Data.

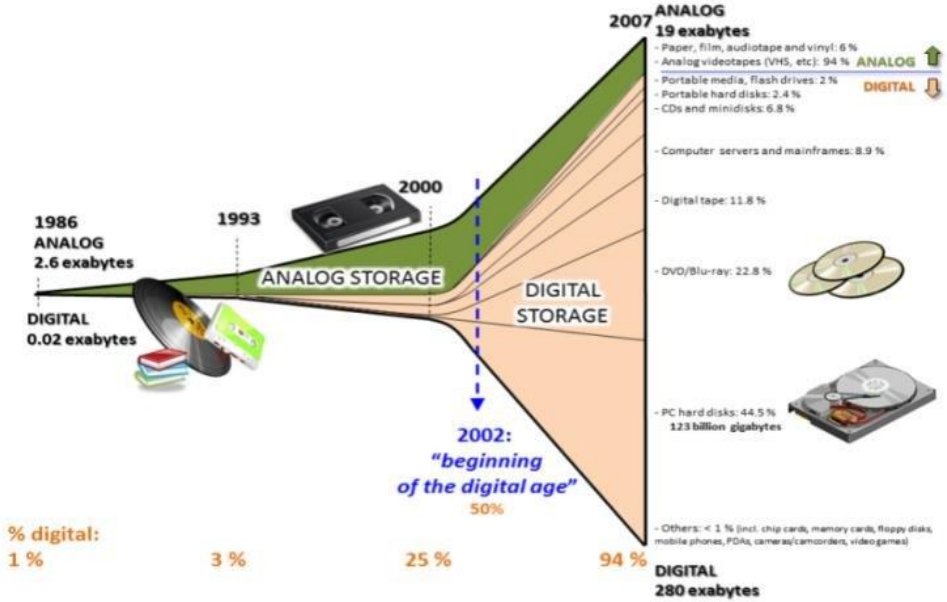
A lo largo de la literatura existente acerca de Big Data, estas **5** características han venido en llamarse "las 5 Vs".

1. **V**olumen.
2. **V**elocidad.
3. **V**ariedad.
4. **V**eracidad.
5. **V**alor.



1.1.1 Volumen

- La primera característica del reto de tratamiento de datos que ha venido en llamarse Big Data es el **volumen** de los mismos, es decir, la gran **cantidad de bytes** de información que los componen.



1.1.1 Volumen

- Para hacernos una idea del volumen de datos que maneja la humanidad, según las predicciones el volumen de datos en el mundo se calculaba en unos **4.4 zettabytes** en 2013, y tiene un crecimiento exponencial según el cual se espera que pueda llegar a los **163 zettabytes** para el año 2025.

1.1.1 Volumen: Procedencia de los datos.

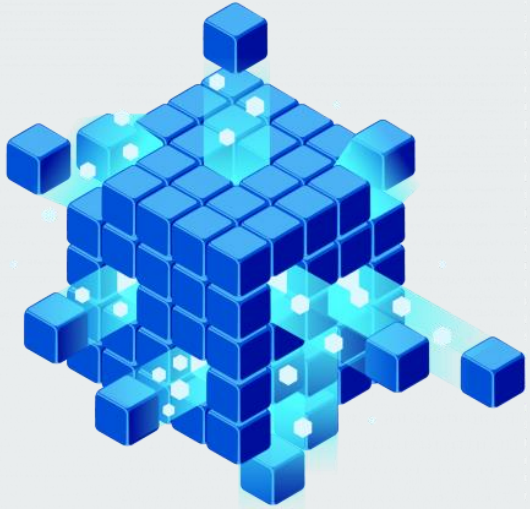
- ¿De dónde vienen todos estos datos?
  - Datos de **usuarios** y/o clientes de instituciones y empresas.
  - Datos generados por **transacciones** (compras, transferencias, ...).
  - Datos adquiridos por **sensores** (de temperatura, de humedad, ...).
  - Datos subidos a **redes sociales** (textos, imágenes, vídeos, ...).
  - Datos relacionados con la **salud** (historiales y pruebas realizadas a pacientes).
  - Datos de **geolocalización** (posicionamiento en cada momento según GPS).
  - Datos guardados en **logs** (de todos los accesos que hacemos a páginas web).
  - Datos producidos por el Internet de las cosas (de los diversos dispositivos **IoT**).
  - Datos producidos por la **genómica** (cada vez que se secuencia un genoma).

1.1.1 Volumen: Procedencia de los datos.

- Datos de **meteorología** (información obtenida por satélites y las predicciones realizadas a partir de la misma).
- Datos producidos por **cámaras** (imágenes estáticas y vídeos producidos).
- Datos producidos por **micrófonos** (grabaciones de sonido producidas).
- Datos de **RFID** (aquellos con los que se tratan al realizar identificación por radiofrecuencia).
- Datos producidos por los sectores **energético e industrial** (toda la información que se genera alrededor de la energía y la industria).
- Datos Open Data (todos los datos abiertos liberados ya sea a nivel gubernamental o no gubernamental).

1.1.2 Velocidad

- No sólo tratamos con una gran cantidad de datos que hay que almacenar y procesar, sino que tales datos a su vez **se siguen produciendo a una gran velocidad**
- Para hacernos una idea, se calcula que en el mundo se generan cada 60 segundos:
  - 350.000 tweets
  - 300 horas de vídeo subidos a YouTube (más los que se suban a otras plataformas)
  - 171 millones de correos electrónicos
  - 330 GBs de información generados por sensores de motores de aviones comerciales
- El problema con respecto a la velocidad no es únicamente el hecho de que el volumen de datos continúe creciendo sin parar, sino lo **rápido que es necesario obtenerlos** y **ser capaces de integrarlos junto con los que ya tenemos**
- De la gran velocidad a la que llegan datos nuevos nacen las estrategias de procesamiento tipo **streaming**

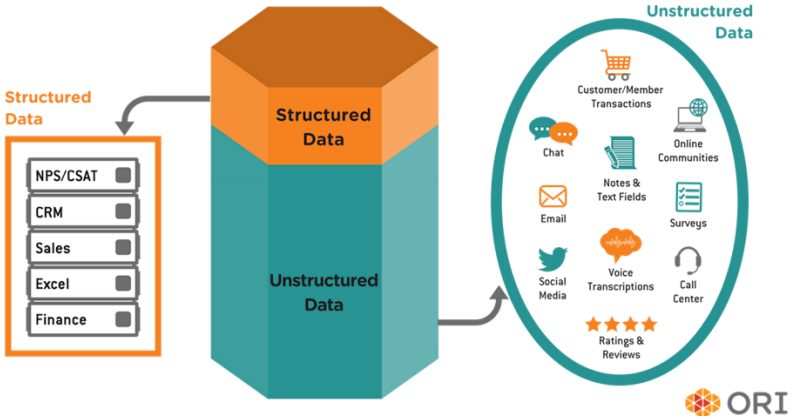


1.1.3 Variedad

- Además de tener que procesar una gran cantidad de datos que se generan cada vez más rápido, existe el problema añadido de la **gran variedad** existente en cuanto a la **representación** de tal información

1.1.3 Variedad

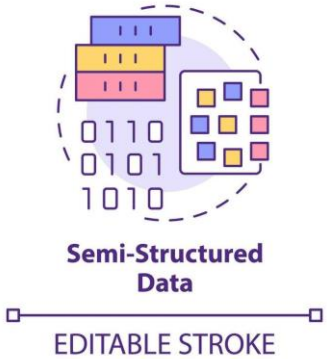
- **Datos estructurados**. Los existentes en registros (filas) de bases de datos (típicamente relacionales), los cuales existen dentro de tablas con un esquema definido que nos indica de qué tipo de datos es cada una de las columnas (entero, decimal, textual, fecha, ...).
- **Datos no estructurados (80%)**. Aquellos que no están regidos por un esquema. Por ejemplo:
  - Vídeos
  - Imágenes
  - Audios



1.1.3 Variedad

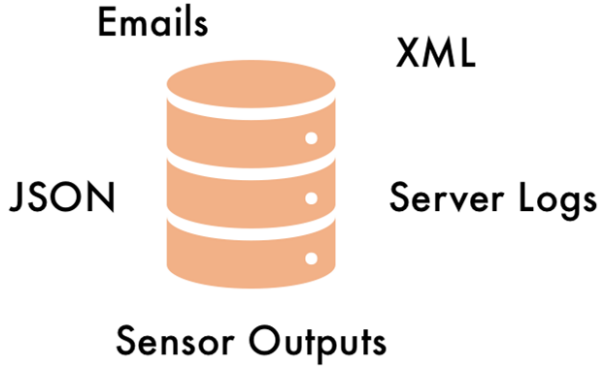
Datos Semiestructurados

- Son datos que siguen una cierta estructura, pero no en forma relacional (no están organizados en tablas con un esquema fijo).
- Se almacenan en archivos de texto con un formato predefinido, lo que permite flexibilidad al guardar información, al tiempo que facilita identificar el significado de cada parte.
- **Ejemplos de formatos**:
  - CSV
  - XML
  - JSON



1.1.3 Variedad

- **Metadatos**. Los metadatos son datos extra (muchas veces generados de forma automática) que se guardan acerca de los propios datos para favorecer su interpretación posterior. Ejemplos de metadatos que pueden acompañar a los datos convencionales son:
  - Información extra sobre su estructura
  - Fuente
  - Autor
  - Fecha de creación
  - Resolución en pixels (si se trata de una imagen o un vídeo)
  - Duración (si se trata de un vídeo)
  - Frecuencia de muestreo (si se trata de un audio)
  - Tipo de compresión



1.1.4 Veracidad

- Un problema extra con el que tenemos que tratar es el hecho de que los datos no siempre cuentan con la calidad deseada o no son totalmente **fieles a la realidad**. Este término está muy relacionado con el concepto de relación señal/ruido en cualquier flujo de información.
  - **El ruido** son datos que no pueden ser convertidos en información (ya sea porque no la contienen o porque ésta está corrupta y es irrecuperable)
  - La señal está constituída por datos que sí pueden ser convertidos en información con sentido

1.1.4 Veracidad

Calidad de los Datos:

- Es crucial conocer las **condiciones** en que se adquirieron los datos para estimar su **veracidad**.
- Se requiere un **procesamiento** específico para corregir problemas y eliminar información inválida.
- Los datos generados **automáticamente** (como transacciones) suelen tener menos **ruido** que los producidos por personas (como posts en blogs).

1.1.5 Valor

- Cómo son de **útiles los datos** para una institución, empresa o persona.
- Tiene mucho que ver con el concepto de veracidad que ya hemos visto, ya que por lo general cuanto **más veraces** (fieles a la realidad) sean los datos, **más valor** se puede obtener de ello.
- También depende en gran medida del **tiempo transcurrido** desde que se produjeron tales datos. Cuanto **más rápido** seamos capaces de hacer llegar el dato desde donde se produce al lugar en el que se toman las decisiones, **más valor** podremos obtener de ellos.
- Es también muy importante que los datos sean lo más **completos** posible (que venga todo lo que necesitamos)

1.1.5 Valor

Si los datos van perdiendo valor con el tiempo y tenemos muchos datos antiguos, ¿merece la pena utilizarlos siempre junto con los más nuevos?

No siempre. Dependerá del caso.

Sobre todo para el análisis descriptivo suele ser útil tener en cuenta datos antiguos para comprobar cuáles son las tendencias (es decir, tener algo con lo que comparar los datos nuevos).

En otros casos, como por ejemplo en la generación de modelos predictivos para intentar acertar con una oferta, lo que queremos es tener cuantos más datos nuevos mejor para así no necesitar usar los más antiguos (ya que los gustos de los usuarios son cambiantes).

1.2 ¿Qué conseguimos gracias a Big Data?

- Las metodologías y tecnologías para Big Data nos permiten realizar diversas **operaciones** con **grandes cantidades de datos**, entre las cuales se encuentran:
  - Capturarlos desde sus orígenes
  - Integrarlos para poderlos almacenar de un modo unificado
  - Almacenarlos de un modo distribuido y replicado, gracias lo cual conseguimos altos valores de disponibilidad
  - Tratarlos de forma distribuida, empleando para ello un alto número de máquinas que los procesan en paralelo
  - Aplicar técnicas de minería de datos (también llamado ciencia de datos cuando esa minería de datos se realiza en ambientes Big Data) para crear modelos predictivos
  - Usar esos modelos para realizar predicciones a utilizar en sistemas automáticos
  - Crear visualizaciones y cuadros de mando usando tanto los propios datos como los modelos creados para así dar soporte a la toma de decisiones

1.2 ¿Qué conseguimos gracias a Big Data?

- El ser capaces de realizar tales operaciones con los datos, nos permiten obtener los siguientes **aportes y beneficios** (entre otros):
  - Generar registros más detallados mediante la integración desde diversas fuentes
  - Optimizar las operaciones de instituciones y empresas
  - Poder actuar de modo inteligente basándonos en la evidencia de los datos
  - Identificar nuevos mercados
  - Realizar predicciones basándose en modelos creados a partir de los datos
  - Detectar casos de fraude e impagos
  - Dar soporte a la toma de decisiones
  - Realizar descubrimientos científicos
  - Ayudar a los médicos a detectar enfermedades en función del historial de los pacientes y las pruebas que se les realizan
  - Crear nuevos fármacos más efectivos y con menos efectos secundarios

1.2.1 Desde los eventos al valor

El tratamiento de los datos a lo largo de diversas **capas de procesamiento sucesivas** nos permite llegar desde los meros **eventos** que se producen en nuestro mundo hasta la sabiduría que necesitamos para obtener **valor** gracias a poder tomar las mejores decisiones.

- **Eventos**: En nuestro mundo se producen eventos constantemente.
  - Una estación meteorológica toma mediciones de temperatura, humedad, presión atmosférica, etc.
  - Una cámara toma imágenes dentro de una fábrica
  - Alguien pide un préstamo
  - Alguien realiza una llamada telefónica
  - Alguien realiza un pago con tarjeta
  - Un hospital realiza una prueba médica a un paciente
  - ...

1.2.1 Desde los eventos al valor

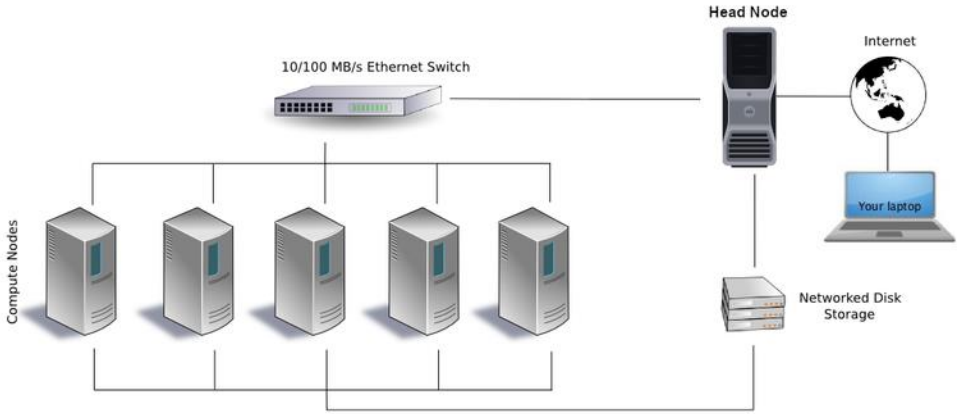
- **Datos**: Los eventos son reflejados de algún modo, generándose de ese modo datos que pueden ser almacenados para su uso posterior
  - Registros de bases de datos
  - Ficheros (en diversos posibles formatos)
- **Información**: Cuando les damos contexto a los datos, organizándolos de algún **modo lógico**, tenemos información
  - Distintos registros referentes a pagos con tarjeta quedan almacenados en una misma tabla
  - Usamos jerarquías de carpetas para organizar distintos ficheros en función de su tipo o significado (fotografías, audios, facturas, ...)
- **Conocimiento**: Si tratamos la información dándole un significado, podemos obtener conocimiento
  - A partir de gran cantidad de datos se generan modelos mediante los cuales se representa la realidad y que pueden ser utilizados para realizar predicciones

1.2.1 Desde los eventos al valor

- **Sabiduría**: Si una vez tenemos conocimiento en forma de modelos predictivos añadimos el entendimiento necesario para saber de qué modo emplearlos. Como resultado obtenemos sabiduría
- **Valor**: La sabiduría por sí misma no genera ninguna acción. Sin embargo, si realizamos acciones basándonos en la sabiduría, esas acciones serán mejores que las que podamos tomar sin basarnos en los datos. La diferencia entre el resultado que podemos obtener basándonos en la sabiduría que producen los datos y el que obtendríamos si no los hubiésemos tenido en cuenta para nada, es el **valor añadido** que conseguimos

2 Clusters de computadoras

- En ambientes de computación, un cluster es un **conjunto de computadoras** (también referenciados como servidores o como nodos) **conectados** entre sí mediante red para **trabajar como una única** unidad resolviendo cargas de trabajo de forma conjunta.
- Históricamente los clusters se construían utilizando computadoras especializadas muy caras. Sin embargo, más adelante han ido apareciendo diversos frameworks o plataformas de computación distribuída que emplean computadoras de uso común (el llamado **commodity hardware**), gracias al considerable aumento de sus prestaciones.



2 Clusters de computadoras

- El uso de clústers nos da una serie de **ventajas** respecto al uso de computadoras de forma individual:
  - Alto rendimiento
  - Alta disponibilidad
  - Equilibrado de carga
  - Escalabilidad

2 Clusters de computadoras

Alto rendimiento:

Dado que cada componente del cluster es una computadora completa, con sus propios recursos (procesador, memoria y almacenamiento), las cargas de trabajo susceptibles de paralelización pueden acelerarse en gran medida **dividiéndolas en subtareas y distribuyéndolas** para que sean ejecutadas en los distintos nodos.

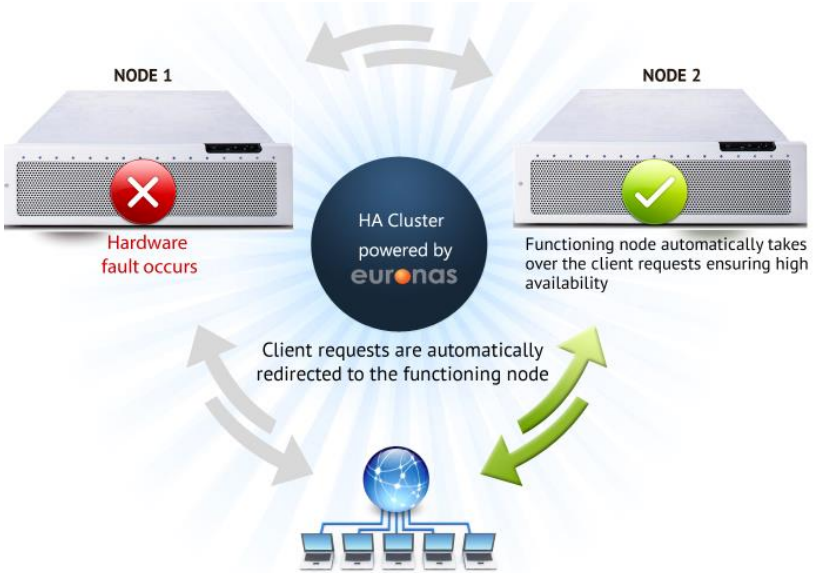
Gracias a esto se pueden resolver problemas muy complejos que no sería posible resolver en un tiempo razonable en una máquina individual por muy potente que ésta sea.

2 Clusters de computadoras

Alta disponibilidad:

Mediante una contínua **monitorización** entre los propios nodos del cluster, se puede detectar la no disponibilidad de un subconjunto de los mismos (ya sea por fallo eléctrico, por avería o por corte de las comunicaciones) y se pueden tomar medidas para que los servicios o datos que hay (o había) en esas máquinas sigan estando disponibles.

- Rearrancando un **nodo caído** o arrancando un nuevo nodo para suplirlo
- Respondiendo las peticiones desde otro nodo del clúster que también contenga una **réplica** de esos datos



2 Clusters de computadoras

Equilibrado de carga:

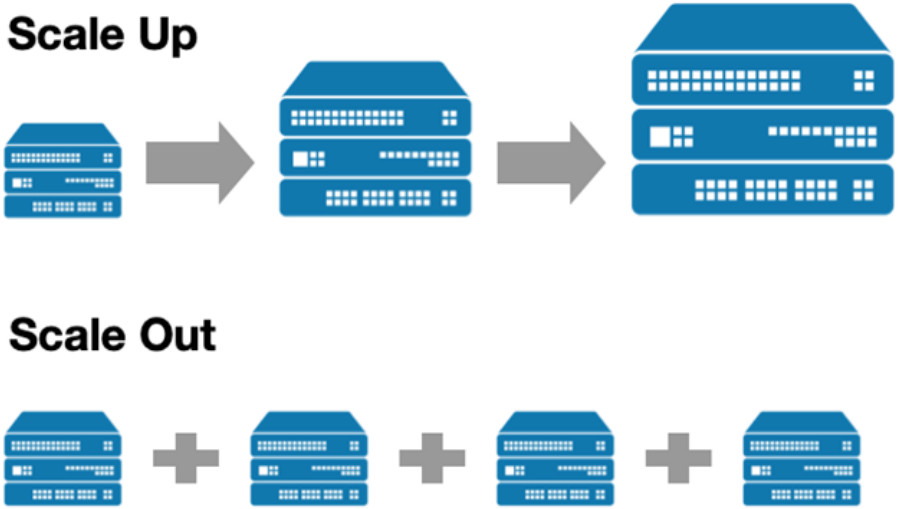
- El equilibrado de carga (o también balance o balanceo) se consigue mediante algoritmos destinados a **distribuir las cargas de trabajo entre los diversos nodos** del clúster para así evitar cuellos de botella.
- Los cuellos de botella se producen cuando el envío de trabajos a nodos sobrecargados aumenta la latencia media.
- Para ello, se realiza una monitorización del estado de carga de cada nodo y se decide para cada paquete de trabajo a qué nodo enviarlo, atendiendo a:
    - El tamaño del trabajo
    - El estado de carga de cada nodo
    - La potencia de procesamiento de cada nodo

2 Clusters de computadoras

Escalabilidad:

- **Clúster escalable:** Compuesto por múltiples nodos, permite mayor potencia de cálculo.
- **Ventaja principal:** La potencia del clúster puede crecer añadiendo nodos.
- **Beneficio para Big Data:** No es necesario hacer una estimación excesiva de recursos al inicio, se puede adaptar el sistema según la demanda.

2 Clusters de computadoras



Tipos de Escalado:

- **Escalado vertical (scale-up):**
  - Aumenta la capacidad de **un solo equipo** (mejor procesador, memoria, etc.).
  - **Limitado** por la capacidad del hardware disponible.
  - No permite escalabilidad real.
- **Escalado horizontal (scale-out):**
  - Aumenta la capacidad añadiendo **más nodos** a un clúster.
  - Permite una **escalabilidad eficiente y sin límites** definidos por el hardware.

CASO PRÁCTICO

3 Conceptos de almacenamiento de datos

- ¡No hay manera! —dice **Pedro**.
- ¿Cómo no va a haber manera? —replica **María**.
- Te digo que no hay manera. La tabla que necesitamos tiene tantos campos y tantos registros que no nos cabe ni en el disco duro del ordenador. ¡Y es el más grande que se puede comprar ahora mismo!
- ¿Y entonces?
- Tendríamos que partir la tabla y guardar los fragmentos en datos en varios ordenadores.
- Claro, ¿y luego cuando tengamos que hacer una consulta cómo sabemos en qué ordenador está justo lo que buscamos? ¿Y si está esparcido por todos los fragmentos?

3 Conceptos de almacenamiento de datos

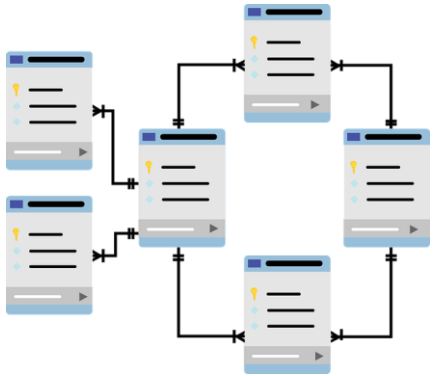
En esta sección vamos a realizar un recorrido a lo largo de una serie de conceptos relacionados con almacenamiento que es importante conocer si vamos a trabajar en entornos Big Data.

Veremos aquí un esquema/resumen para que podamos tener una vista general:

- **Base de Datos Relacional:**
  - **Más utilizada** a nivel mundial, pero **NO escalable** para grandes volúmenes de datos en Big Data.
- **Dataset:**
  - Conjunto de datos, a menudo de tamaño **enorme.**
- **Almacén de Datos:**
  - Sistema especializado para **almacenar datos**, generalmente utilizado para **análisis**.
- **ACID:**
  - Conjunto de propiedades que las bases de datos deben cumplir para realizar **transacciones seguras** (Atomicidad, Consistencia, Aislamiento, Durabilidad).
- **Teorema CAP:**
  - Explica las tres propiedades de bases de datos distribuidas: **Consistencia, Disponibilidad y Tolerancia a Particiones**. No es posible conseguir las tres al mismo tiempo.
- **BASE:**
  - Principio de diseño de bases de datos distribuidas que prioriza la **disponibilidad** sobre la consistencia total.

3.1 Base de Datos Relacional

- **Estructura**: Almacena datos en **tablas** con **filas** (registros) y **columnas** (atributos).
- **Esquema**: Define los tipos de datos (entero, texto, fecha, etc.) para cada columna, garantizando **uniformidad**.
- **Altas prestaciones**: Los motores de bases de datos relacionales permiten realizar búsquedas rápidas gracias a los **índices** que aceleran la búsqueda y las **relaciones** entre tablas.
- **Relaciones**: Permiten consultas que involucran varias tablas, de ahí el término "relacional".



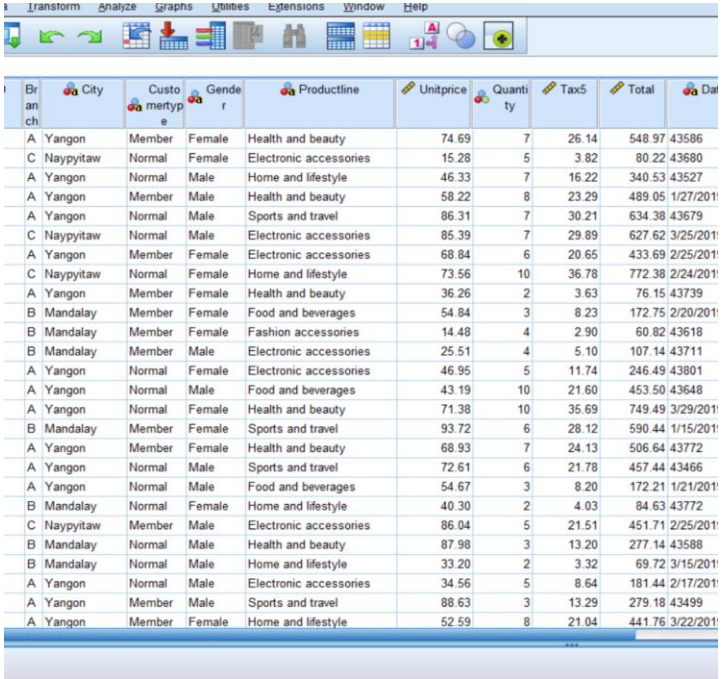
➢ Esta organización permite un **rendimiento óptimo** en la gestión de datos estructurados, pero presenta limitaciones para grandes volúmenes en Big Data.

3.1 Base de Datos Relacional

- **Alto rendimiento para transacciones**, pero limitado por el tamaño de las tablas.
- Escalan de forma **vertical**: mejorando un solo servidor.
- **Problema para Big Data**: Este enfoque se vuelve un **cuello de botella**, ya que no siempre es posible mejorar indefinidamente el hardware del servidor.

➢ En entornos Big Data, se necesitan soluciones que permitan un **escalado horizontal** para manejar volúmenes masivos de datos, algo que las bases de datos relacionales no ofrecen.

3.2 Dataset



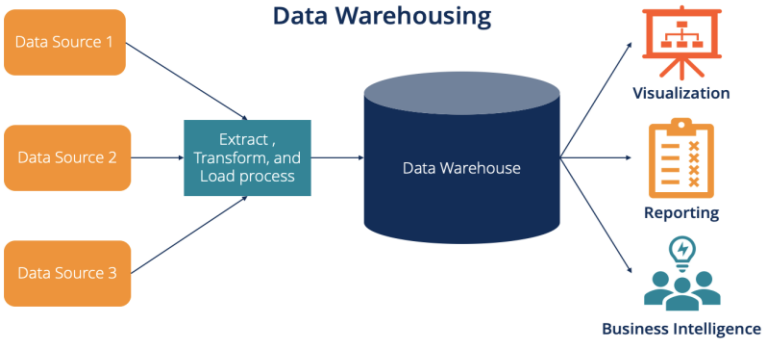
Br an ch	City	Custo mertyp e	Gende r	Productline	Unitprice	Quanti ty	Tax5	Total	Da
A	Yangon	Member	Female	Health and beauty	74.69	7	26.14	548.97	43586
C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82	80.22	43680
A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.22	340.53	43527
A	Yangon	Member	Male	Health and beauty	58.22	8	23.29	489.05	1/27/201
A	Yangon	Normal	Male	Sports and travel	86.31	7	30.21	634.38	43679
C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.89	627.62	3/25/201
A	Yangon	Member	Female	Electronic accessories	68.84	6	20.65	433.69	2/25/201
C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.78	772.38	2/24/201
A	Yangon	Member	Female	Health and beauty	36.26	2	3.63	76.15	43739
B	Mandalay	Member	Female	Food and beverages	54.84	3	8.23	172.75	2/20/201
B	Mandalay	Member	Female	Fashion accessories	14.48	4	2.90	60.82	43618
B	Mandalay	Member	Male	Electronic accessories	25.51	4	5.10	107.14	43711
A	Yangon	Normal	Female	Electronic accessories	46.95	5	11.74	246.49	43801
A	Yangon	Normal	Male	Food and beverages	43.19	10	21.60	453.50	43648
A	Yangon	Normal	Female	Health and beauty	71.38	10	35.69	749.49	3/29/201
B	Mandalay	Member	Female	Sports and travel	93.72	6	28.12	590.44	1/15/201
A	Yangon	Member	Female	Health and beauty	68.93	7	24.13	506.64	43772
A	Yangon	Normal	Male	Sports and travel	72.61	6	21.78	457.44	43466
A	Yangon	Normal	Male	Food and beverages	54.67	3	8.20	172.21	1/21/201
B	Mandalay	Normal	Female	Home and lifestyle	40.30	2	4.03	84.63	43772
C	Naypyitaw	Member	Male	Electronic accessories	86.04	5	21.51	451.71	2/25/201
B	Mandalay	Normal	Male	Health and beauty	87.98	3	13.20	277.14	43588
B	Mandalay	Normal	Male	Home and lifestyle	33.20	2	3.32	69.72	3/15/201
A	Yangon	Normal	Male	Electronic accessories	34.56	5	8.64	181.44	2/17/201
A	Yangon	Member	Male	Sports and travel	88.63	3	13.29	279.18	43499
A	Yangon	Member	Female	Home and lifestyle	52.59	8	21.04	441.76	3/22/201

- **Dataset**: Conjunto de datos con una **relación** significativa entre ellos.
  - Ejemplos:
    - Colección de **tweets**.
    - Colección de **posts**.
    - Colección de **imágenes**.
    - **Registros** de bases de datos relacionales o no relacionales.
    - Medidas de una **estación meteorológica**.
- **Formatos** de almacenamiento:
  - **Texto plano**: CSV, XML, JSON, o sin formato específico.
  - **Tablas de bases de datos**.
  - **Ficheros multimedia**: imágenes, vídeos, audios.

➢ En resumen, un dataset puede abarcar distintos tipos de datos y formatos, y es la unidad básica con la que se trabaja en el análisis y almacenamiento de grandes volúmenes de información en Big Data.

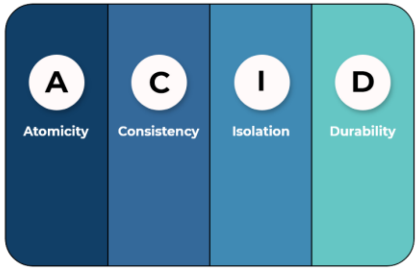
3.3 Almacén de Datos

- Un **almacén de datos** o **data warehouse** es un repositorio central que almacena datos tanto actuales como históricos a nivel institucional o empresarial.
- Está diseñado principalmente para **inteligencia de negocio (BI)** y **consultas analíticas**, y no para transacciones.
- Los datos se cargan periódicamente a través de procesos ETL (Extracción, Transformación y Carga), ofreciendo una **instantánea** de la información en un momento dado.



3.4 ACID

- El principio **ACID** es fundamental para el diseño de bases de datos transaccionales, garantizando que las transacciones sean **seguras y confiables**.
- ACID consta de cuatro propiedades esenciales de **obligado cumplimiento:**
  - Atomicidad (**a**tomicity).
  - Consistencia (**c**onsistency).
  - Aislamiento (**i**solation).
  - Durabilidad (**d**urability).



- Este tipo de gestión realiza un **control pesimista de la concurrencia**, dando por hecho que cualquier problema que pueda ocurrir ocurrirá tarde o temprano por poco probable que sea (aplicando la Ley de Murphy).

- Para conseguirlo, el motor de la base de datos **bloquea** registros individuales e incluso tablas completas en determinados momentos **para asegurar** que la **consistencia** se mantiene en todo momento.

3.4 ACID

- **Atomicidad**:
  - Una transacción se ejecuta **todo o nada**.
  - Si una parte de la transacción falla, se **revierte** todo lo que ya se haya realizado.

- **Ejemplo**:
  - Si una transacción intenta insertar dos registros, y la primera inserción tiene éxito pero la segunda falla, **ningún cambio** será consolidado. La base de datos volverá a su **estado inicial**.

➢ Este principio asegura que la base de datos no quede en un estado **inconsistente** si ocurre un fallo durante una transacción.

3.4 ACID

- **Consistencia**:
  - La base de datos se mantiene en un **estado válido** en todo momento.
  - Se verifica que los datos cumplan con los **esquemas** y **reglas** antes de realizar cualquier escritura.

- **Resultado**: Después de una transacción exitosa, la base de datos sigue siendo **consistente**, es decir, todos los datos siguen cumpliendo las reglas de integridad definidas.

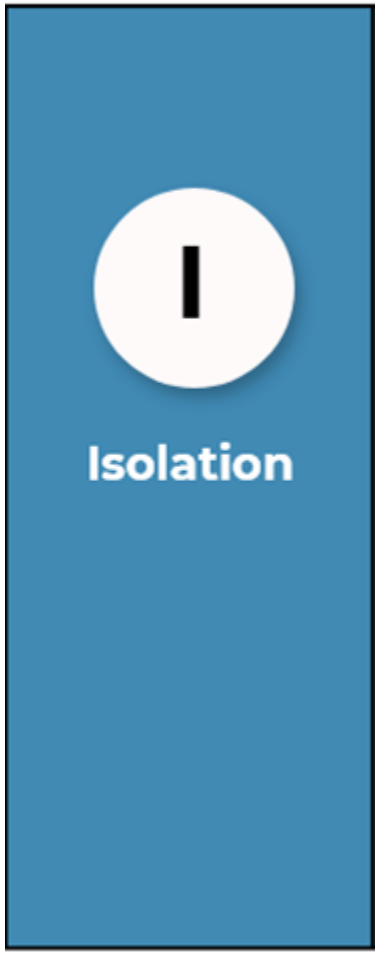
➢ Este principio asegura que los datos siempre sean **coherentes** y que no se permita la escritura de datos inválidos o incorrectos.

3.4 ACID

- **Aislamiento**:
  - Los cambios de una transacción no son visibles para otros procesos hasta que la transacción se haya **completado**.

- **Ejemplo**:
  - Si una transacción inserta dos registros, otros usuarios o procesos no verán solo uno de los registros intermedios. Verán **ambos** registros si la transacción termina, o **ninguno** si aún está en proceso.

➢ Este principio evita **lecturas parciales** o inconsistentes durante la ejecución de una transacción, asegurando que cada proceso vea los datos en un estado **coherente**.

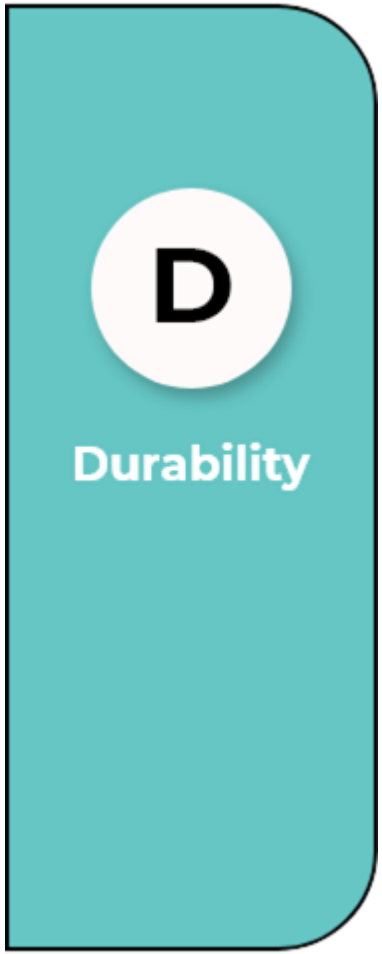


3.4 ACID

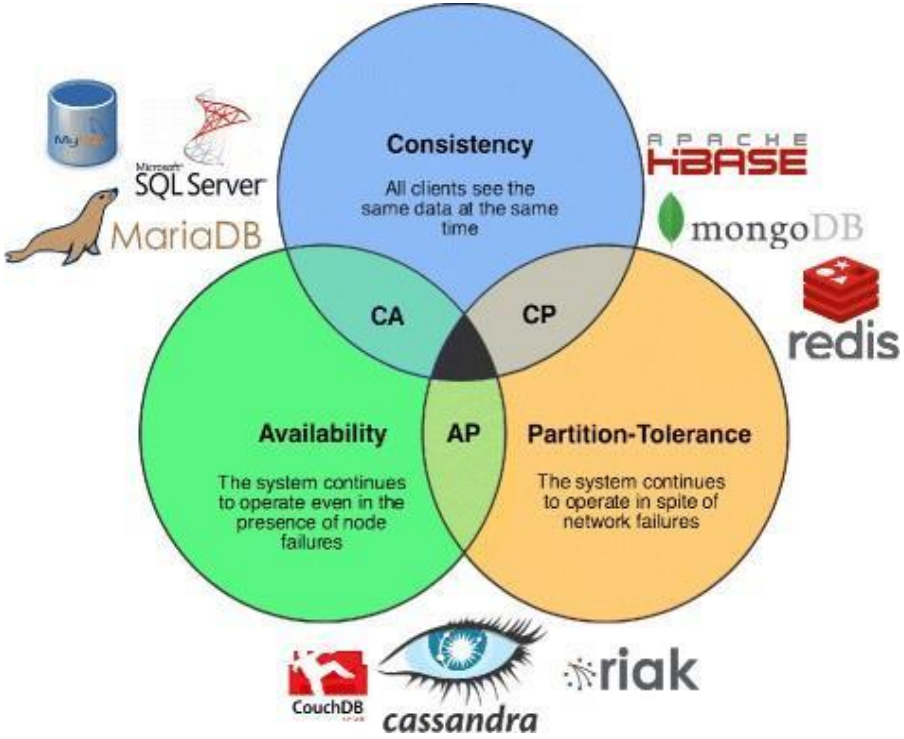
- **Durabilidad**:
  - Los cambios en la base de datos son **permanentes** y no se perderán tras un fallo del sistema.
  - Los datos se guardan en **almacenamiento no volátil** (discos duros o SSD).

- **Impacto en el rendimiento**:
  - El acceso a almacenamiento no volátil es alrededor de **100 veces más lento** que el acceso a memoria, lo que **limita** la velocidad de las bases de datos que siguen el modelo ACID.

➢ Este principio es clave para garantizar que los datos permanezcan seguros y no se pierdan en caso de fallos, aunque introduce un compromiso en la **velocidad** de las operaciones de escritura.



3.5 Teorema CAP



- **Teorema CAP**:
  - Una base de datos distribuida puede cumplir solo con **dos** de las siguientes tres propiedades:
    - **C**onsistencia (C): Todos los nodos ven la misma información al mismo tiempo.
    - **A**vailability (A): Disponibilidad. El sistema siempre responde, incluso si algunas partes fallan.
    - **P**artition Tolerance (P): El sistema sigue funcionando a pesar de la pérdida o retraso en la comunicación entre nodos.
- **Implicación**: Debes elegir entre:
  - **C + A**: Consistencia y disponibilidad, pero no tolerancia a fallos de red.
  - **C + P**: Consistencia y tolerancia a particiones, pero no siempre disponible.
  - **A + P**: Disponibilidad y tolerancia a particiones, pero no consistencia total.

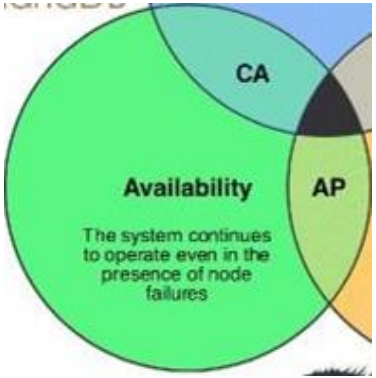
➢ Al diseñar una base de datos distribuida, es fundamental decidir **qué propiedad sacrificar** según las necesidades del sistema.

3.5 Teorema CAP

- **Consistencia (C)**:
  - **Lecturas** siempre muestran el estado más reciente después de la última **escritura**.
  - Si no es posible devolver el estado actualizado, se debe devolver un **error**, pero **nunca** información desfasada.

➢ Este principio asegura que los datos siempre estén **actualizados** y **sin inconsistencias**, aunque puede afectar la disponibilidad del sistema si se prioriza esta propiedad.

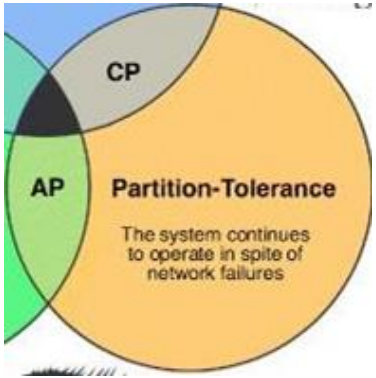
3.5 Teorema CAP



- **Disponibilidad (A):**
  - **Siempre** se da una respuesta válida, sin errores.
  - No garantiza que la respuesta corresponda al **estado más reciente** de los datos.

➢ Este principio prioriza que el sistema esté **siempre accesible** para los usuarios, incluso si los datos pueden no estar completamente actualizados.

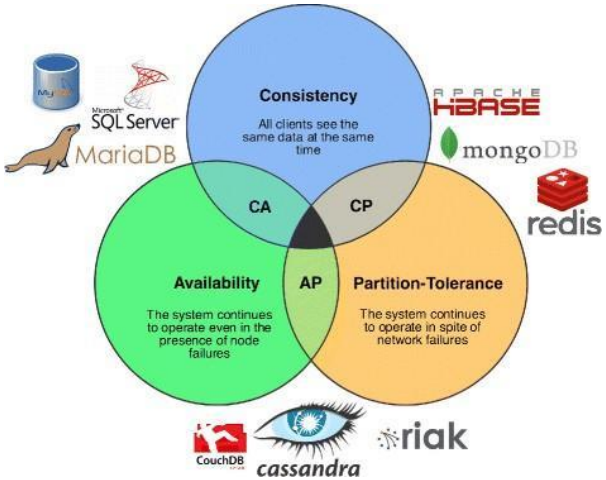
3.5 Teorema CAP



- **Tolerancia al particionamiento (P):**
  - El sistema sigue **operando** y dando respuestas a pesar de fallos o pérdida de comunicación entre nodos.
  - Las lecturas pueden no incluir los datos **más recientes** escritos en otros nodos debido a la falta de sincronización.

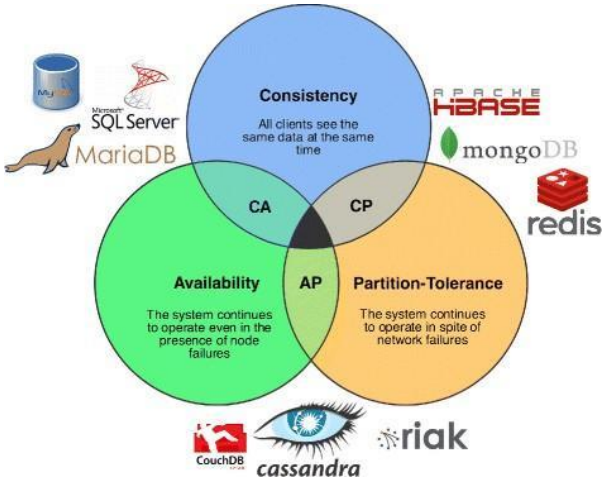
➢ Este principio asegura la **resiliencia** del sistema frente a fallos en la red, aunque puede afectar la consistencia de los datos disponibles en diferentes nodos.

3.5 Teorema CAP - Escenarios



- **Escenario 1:** Consistencia (C) y Disponibilidad (A):
  - Los nodos **deben comunicarse** para garantizar que las lecturas sean consistentes y disponibles, pero esto **rompe la tolerancia al particionamiento (P).**
- **Escenario 2:** Consistencia (C) y Tolerancia al Particionamiento (P):
  - Los nodos **mantienen la consistencia** a pesar del particionamiento, pero durante ese tiempo **no pueden estar disponibles (A)** para recibir o procesar solicitudes.
- **Escenario 3:** Disponibilidad (A) y Tolerancia al Particionamiento (P):
  - El sistema **permanece disponible** y funciona incluso durante fallos en la red, pero **es necesario aceptar lecturas inconsistentes** ya que los nodos no pueden comunicarse para asegurar la **consistencia (C).**

3.5 Teorema CAP - Conclusión



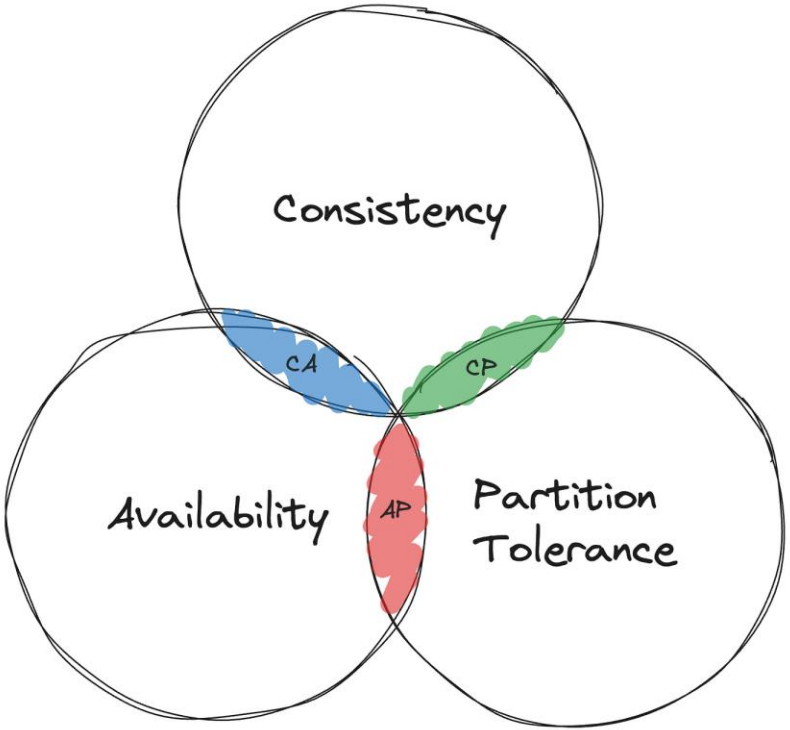
Es **imposible** cumplir con **consistencia**, **disponibilidad** y **tolerancia al particionamiento** simultáneamente. En sistemas distribuidos, se debe **priorizar** qué dos propiedades son más importantes según las necesidades del sistema, aceptando que la tercera se verá comprometida.

3.5 Teorema CAP

- Dado que los cortes de comunicación entre nodos pueden ocurrir en cualquier momento, muchas instituciones o empresas **priorizan la tolerancia al particionamiento (P)** para que su base de datos distribuida siga funcionando.
- En estos casos, se enfrentan a una decisión clave dentro del teorema CAP: deben elegir entre **consistencia (C)** o **disponibilidad (A)**, dependiendo de las necesidades del sistema.

3.5 Teorema CAP

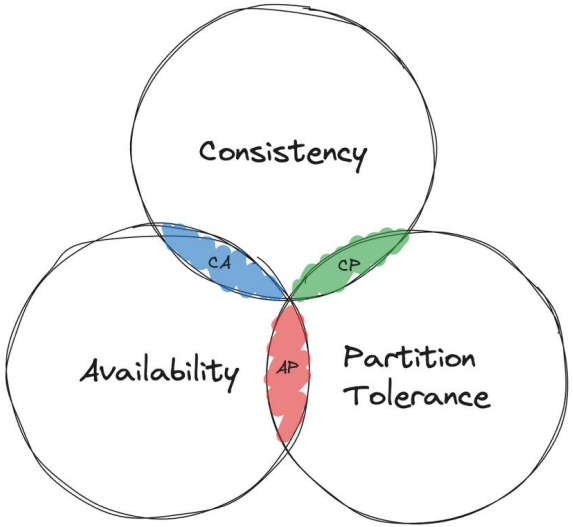
Opciones si se prioriza la tolerancia al particionamiento (P)



3.5 Teorema CAP

Opción 1: Disponibilidad (A) y Tolerancia al Particionamiento (P):

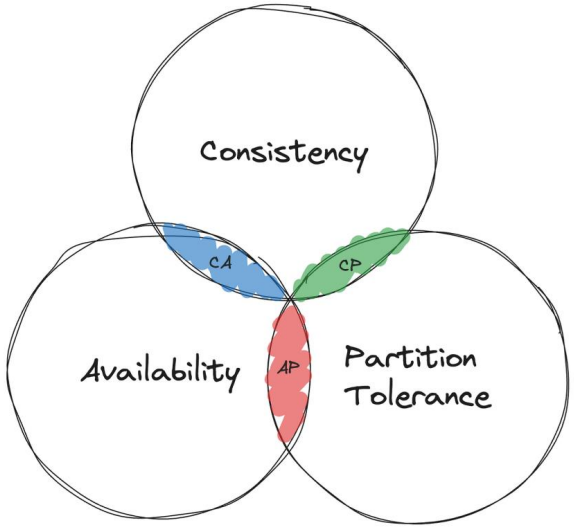
- El sistema **permanece disponible** incluso cuando hay particionamiento.
- **Sacrificio**: No se garantiza la **consistencia** en todas las lecturas. Las lecturas pueden devolver datos **desactualizados** o **inconsistentes**.
- **Aplicación**: Es ideal cuando la **alta disponibilidad** es crucial, como en aplicaciones en tiempo real (por ejemplo, redes sociales o sistemas de e-commerce).



3.5 Teorema CAP

Opción 2: Consistencia (C) y Tolerancia al Particionamiento (P):

- Se garantiza que las lecturas siempre reflejan los datos **más recientes y consistentes**.
- **Sacrificio**: El sistema puede estar **no disponible** durante particionamientos hasta que los nodos recuperen la comunicación y alcancen un estado consistente.
- **Aplicación**: Es adecuada cuando la **consistencia estricta** es esencial, como en sistemas financieros o de contabilidad, donde la precisión de los datos es prioritaria sobre la disponibilidad inmediata.



3.6 BASE

BASE es un principio de diseño de bases de datos **basado** en las restricciones impuestas por el teorema **CAP**, y típicamente empleado por muchas implementaciones de bases de datos distribuídas.

El significado del acrónimo es:

- Básicamente disponible (**b**asically available)
- Estado blando (**s**oft state)
- Consistencia eventual (**e**ventual consistency)

Una base de datos que conforme a la filosofía BASE prefiere la disponibilidad antes que la consistencia (es decir, desde el punto de vista del teorema CAP es A+P)

3.6 BASE

Básicamente disponible:

- Significa que la base de datos **siempre responde a las solicitudes recibidas**, ya sea con una respuesta **exitosa o** con una notificación de **error**, aún en el caso de que se produzca particionamiento entre los nodos (que algunos de ellos caigan o no están accesibles mediante red).
- En ocasiones eso puede significar recibir lecturas desde nodos que no han recibido la última escritura, por lo que el resultado puede **no ser consistente.**

3.6 BASE

- **Estado blando:**
  - La base de datos puede estar en un **estado inconsistente** durante una lectura.
- **Lecturas variables:**
  - Dos lecturas consecutivas pueden devolver **resultados distintos**, aun cuando no haya habido **nuevas escrituras.**
- **Probabilidad de estado final:**
  - Cada lectura tiene solo una **cierta probabilidad** de mostrar el estado final y actualizado de la base de datos.
  - Las **escrituras recientes** pueden no haberse consolidado en el nodo donde se realiza la lectura.

3.6 BASE

Eventualmente consistente:

Significa que **tras cada escritura, la consistencia de la base de datos sólo se alcanza una vez el cambio ha sido propagado a todos los nodos** (de ahí que la consistencia sea eventual en lugar de segura). Durante el tiempo que tarda en producirse la consistencia, observamos un estado blando de la base de datos.

3.6 BASE

Dado que las bases de datos distribuídas que emplean la filosofía BASE dan prioridad a la disponibilidad a costa de la consistencia, ¿son una buena elección para uso transaccional?

No, ya que para usos transaccionales la consistencia es una característica obligatoria.

CASO PRÁCTICO

4. Conceptos de procesamiento de datos

En 1995 **Sega** lanzó la **Sega Saturn** (equipada con 2 procesadores Hitachi SH2 de 32 bits a 28.6 MHz) para competir con la **Sony PlayStation** (equipada con un único procesador de 32 bits a 33.8 MHz).

Sobre el papel parecía que la Sega Saturn tenía toda la ventaja en términos de hardware gracias a la suma de sus dos procesadores, que quizás podrían tener un rendimiento equivalente a 28.6 * 2 = 57.2 Mhz, muy por encima de la velocidad de su rival.

Sin embargo fue precisamente esta arquitectura multiprocesador una de las principales causas de su fracaso.

Por un lado, el contar con dos procesadores hizo mucho más complejo para los desarrolladores el crear juegos para ella. En lugar de estar únicamente concentrados en crear grandes títulos, tenían que dedicar parte del tiempo a encontrar el modo de distribuir el trabajo entre ambos procesadores, lo cual no era tarea sencilla y además producía errores de software extra que debido a la ejecución en paralelo eran muy complicados de detectar. Por esa razón muchos estudios optaban por desarrollar el juego empleando un único procesador.

Por otro lado, incluso encontrando el modo de paralelizar el juego, el rendimiento final no era el equivalente al de un único procesador el doble de rápido ni mucho menos. Ello se debía a que era imposible tener siempre ambos procesadores a pleno rendimiento (ya que no había modo de conseguir un balanceo de carga perfecto), y aparte en muchas ocasiones uno de ellos tenía que quedar parado a la espera de los resultados del otro. Además había que añadir la sobrecarga extra que conlleva la propia comunicación de información entre los procesadores.

Por último, era mucho más caro producir una videoconsola con 2 procesadores (que eran comprados a Hitachi) que si hubiese tenido sólo uno.

4. Conceptos de procesamiento de datos

En esta sección vamos a realizar un recorrido a lo largo de una serie de conceptos relacionados con **procesamiento de datos** que es importante conocer si vamos a trabajar en entornos Big Data.

4. Conceptos de procesamiento de datos

Veremos aquí un esquema/resumen para que puedas tener una vista general:

- **Procesamiento en paralelo:** Distintos procesos dentro del mismo procesador.
- **Procesamiento en distribuido:** Distintos procesos para un mismo trabajo ejecutándose en distintas máquinas.
- **Estrategias de procesamiento de datos:** Cómo trabajamos con datos según el tipo de actividad que vayamos a realizar.
- **OLTP:** Procesamiento transaccional.
- **OLAP:** Procesamiento para analítica.
- **Principio SCV:** Un principio que nos dice qué propiedades podemos conseguir en un sistema de procesamiento distribuido.

4.1 Procesamiento en paralelo I

- **Multitarea**: Los sistemas operativos multitarea han existido mucho antes de los procesadores multihilo o placas multiprocesador.

- Utilizan un **gestor de procesos** que distribuye el **tiempo de ejecución** entre los procesos, asignando ventanas de tiempo de **milisegundos** a cada uno.

➢ La multitarea inicial se logró a nivel **software**, sin necesidad de hardware especializado en paralelo.

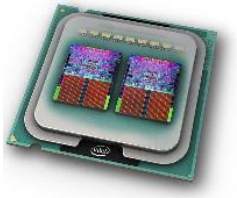
4.1 Procesamiento en paralelo II

- **Placas multiprocesador:**
  - Primer avance para realizar **más de una tarea** realmente al mismo tiempo en hardware.

- **Procesadores multihilo y multinúcleo:**
  - Permiten ejecutar varios hilos o tareas simultáneamente, llevando la **multitarea real** al usuario convencional.

➢ Estos avances permitieron una **verdadera multitarea**, haciendo posible el procesamiento en paralelo tanto a nivel de hardware como de software.

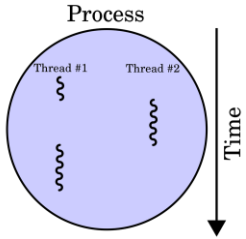
4.1 Procesamiento en paralelo III



Multinúcleo:

El **procesador** contiene **varios núcleos**, cada uno de ellos con una CPU completa.

4.1 Procesamiento en paralelo IV



Multihilo:

Caso en el que una CPU está diseñada de modo que **es capaz de atender a más de un hilo de ejecución** (por lo general 2) permitiendo al segundo utilizar recursos que en ese momento no esté utilizando el primero.

Por ejemplo, el segundo proceso puede realizar una multiplicación de dos valores mientras el primero no está utilizando el recurso necesario para multiplicar porque se encuentra cargando un dato desde memoria.

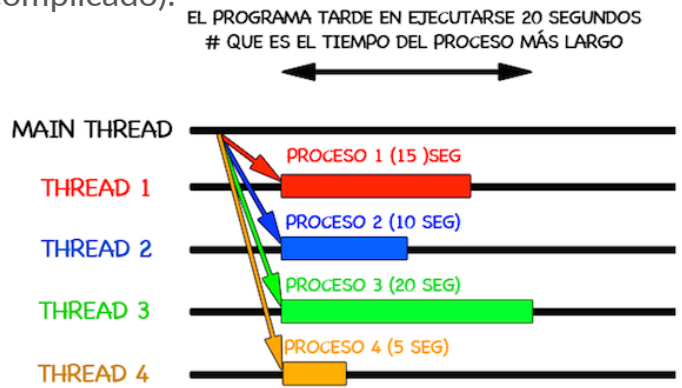
Debido a que en ocasiones ambos procesos necesitan usar el mismo recurso (por ejemplo si ambos necesitan multiplicar en un determinado momento), habrá fracciones de tiempo en los que uno de ellos queda parado a la espera de que el otro termine de utilizar el recurso.

Por esa razón, un procesador comercializado como "de 2 núcleos y 4 hilos" no llega a ser equivalente a un procesador de 4 núcleos físicos.

4.1 Procesamiento en paralelo V

Cuando las **tareas** que se están ejecutando son **independientes** (por ejemplo reproducir un audio, grabar vídeo con una webcam y ejecutar un editor de textos), **no existe ningún problema de paralelización** entre ellas.

El **problema** en cuanto a la paralelización aparece **cuando tenemos una tarea muy compleja** (por ejemplo un análisis sobre una gran cantidad de datos, lo cual es el típico ejemplo de trabajo en ambientes Big Data), y necesitamos dividirla en distintas subtareas **independientes** (lo cual no siempre es posible, y aun siendo posible en ocasiones es muy complicado).

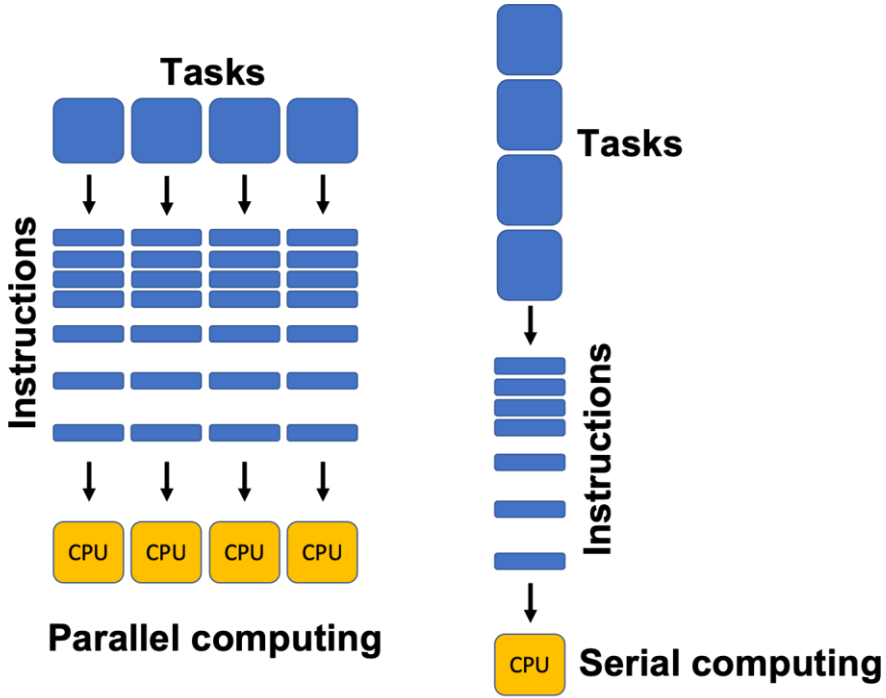


4.1 Procesamiento en paralelo VI

Tarea paralelizable:

Si nos piden sumar mil billones de números aprovechando la potencia de un procesador multinúcleo, podemos separar esos números en tantos paquetes como núcleos y ejecutar un proceso para cada uno de ellos.

Cada proceso realiza la suma de los números del paquete recibido y le devuelve el resultado a otro proceso al que ya sólo le queda sumar esos resultados parciales para obtener el resultado final.



4.1 Procesamiento en paralelo VII

Tarea no paralelizable:

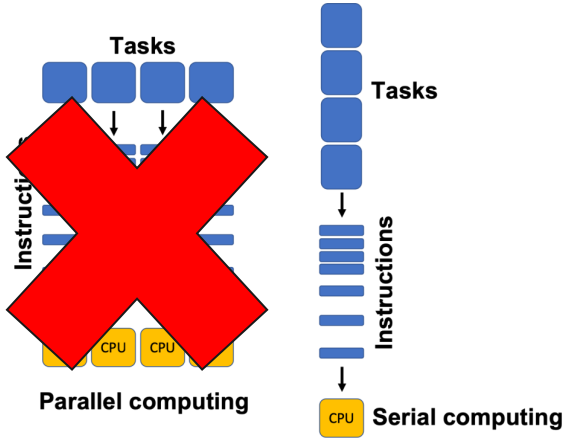
Operación secuencial con mil billones de números:

- **Regla**:
  - Si el resultado parcial es **par**, sumar el número siguiente.
  - Si el resultado parcial es **impar**, restar el número siguiente.

Problema de paralelización:

- Cada nuevo paso **depende** del resultado parcial del paso anterior.
- Los núcleos no pueden trabajar en paralelo sin **esperar** al resultado del núcleo anterior.

➢ No hay un modo eficiente de **dividir el trabajo** entre varios núcleos porque cada núcleo necesita el **resultado previo** para continuar, lo que genera **dependencias secuenciales** y limita el procesamiento paralelo.



4.1 Procesamiento en paralelo VIII

Pero entonces, si sólo tenemos un único procesador que no es multihilo y el gestor de procesos del sistema operativo reparte tiempos entre los procesos, ¿podemos decir que está realizando varias tareas al mismo tiempo?

No. En este caso es una **multitarea simulada,** en el sentido de que gracias a que las ventanas temporales son de milisegundos, el usuario humano tiene la impresión de que su ordenador está haciendo varias cosas a la vez (aunque realmente no sea así).

4.2 Procesamiento distribuido

Procesamiento distribuido:

- Realizado en **distintas máquinas** dentro de un clúster.
- Cada máquina puede tener un **procesador multinúcleo** para ejecutar en paralelo.

Doble nivel de paralelismo:

- **Distribuido en nodos** y **paralelo en procesadores**.

Facilidad de gestión:

- **Frameworks de Big Data** automatizan la gestión de este paralelismo doble, haciéndolo **transparente** para el desarrollador.

4.2 Procesamiento distribuido

- **Comunicación entre procesos**:
  - Para trabajar de forma conjunta, los procesos deben **intercambiar datos** parciales.
- **Impacto en el tiempo**:
  - La **ubicación** de los procesos (local o en otra máquina) afecta el **tiempo** de comunicación.

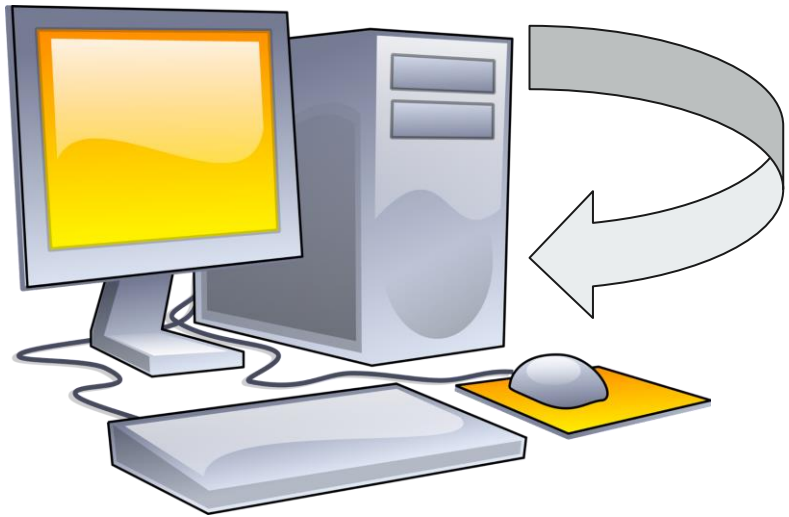
➢ Al diseñar sistemas de **Big Data**, es crucial considerar el **tiempo de comunicación** entre procesos según su ubicación en el clúster.

4.2 Procesamiento distribuido

Comunicación dentro de la misma máquina:

Es el caso de **comunicación más rápido** posible, ya que los datos no necesitan ser enviados por red sino que los procesos pueden intercambiarlos a través de recursos residentes en la propia máquina.

- Memoria RAM
- Sistema de ficheros
- Base de datos (útil pero lento)



4.2 Procesamiento distribuido

Comunicación en red:

- Es varias órdenes de magnitud **más lenta** que la comunicación local en una máquina.
- Implica enviar datos al **interfaz de red**, utilizar un protocolo de comunicación y recibirlos en otra máquina a través de su interfaz.

Big Data:

- En estos entornos, el uso de **múltiples nodos** en un clúster hace necesaria esta comunicación.

Optimización:

- La manera más rápida de comunicar datos entre máquinas es si los nodos están conectados al **mismo switch**, ya que reduce el número de saltos entre elementos de red.

4.2 Procesamiento distribuido

Saltos entre switches:

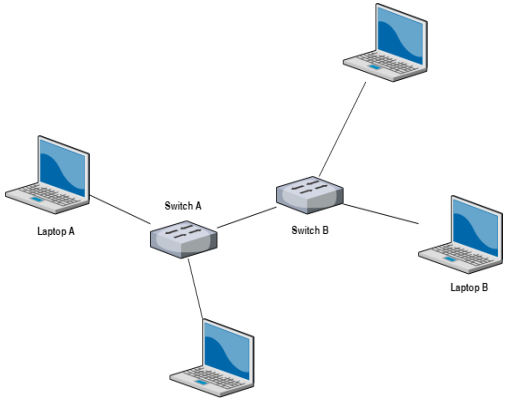
- Cuantos más saltos entre **switches interconectados**, más **lenta** es la comunicación.

Organización del CPD:

- Los servidores dentro de un mismo **rack** suelen estar conectados a un mismo switch.
- Existen jerarquías de switches para conectar **racks** entre sí.

Gestión del clúster:

- El software del clúster se configura con la **jerarquía de switches** para tomar las mejores decisiones sobre **emplazamiento de datos** y procesos.
- El objetivo es **minimizar los saltos** entre switches y mejorar el rendimiento.



4.2 Procesamiento distribuido

Origen y destino de los datos:

- Los datos procesados en un clúster suelen originarse **fuera del CPD**, por lo que deben ser **transferidos** hacia el CPD.
- Del mismo modo, los resultados generados por el clúster frecuentemente deben ser **extraídos** del CPD para su uso externo.

Velocidad de comunicación:

- La comunicación fuera del CPD es **mucho más lenta** que la comunicación interna, que se realiza a través de switches de alta velocidad.
- Este es el tipo de **comunicación más lenta** debido a factores como el uso de redes externas y distancias más grandes.

4.3 Estrategias de procesamiento de datos

En entornos Big Data se emplean distintas estrategias a la hora de procesar los datos, las cuales se escogen según la cantidad y naturaleza de los mismos, así como de las necesidades de la actividad que se esté realizando.

- **Por lotes** (del inglés batch)
- **Transaccional**
- **En tiempo real** (del inglés realtime)
- **Streaming** (anglicismo usado sin traducir al castellano)

4.3 Estrategias de procesamiento de datos

- **Procesamiento por lotes**:
  - No requiere producir respuestas **inmediatas.**
  - Se ejecuta en **grandes cantidades de datos** de forma acumulada.
  - Puede tardar **horas o días** en completarse.

- **Aplicación:**
  - Principalmente usado en **análisis de grandes volúmenes** de datos, trabajando con todos los datos disponibles para una tarea específica.

  ➢ El procesamiento por lotes es ideal para **analítica en profundidad** cuando no se necesita un resultado inmediato, permitiendo el análisis de grandes conjuntos de datos con mayor precisión.

4.3 Estrategias de procesamiento de datos

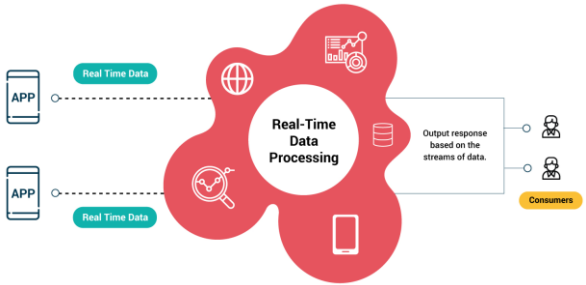
- **Procesamiento transaccional**:
  - **Respuestas inmediatas**: Tiempos de respuesta muy cortos, preferiblemente **menos de un segundo**.
  - Se utiliza en el manejo de **transacciones**.

- **Restricciones**:
  - No se puede procesar **grandes volúmenes de datos** debido a las limitaciones de tiempo.

➢ El procesamiento transaccional es clave en situaciones donde la **rapidez** es fundamental, como en sistemas de **pago** o **bases de datos financieras**, donde los retrasos no son aceptables.

4.3 Estrategias de procesamiento de datos

- **Procesamiento en tiempo real**:
  - **Respuestas rápidas**, para que el usuario humano pueda consultar **estadísticas** o resultados de datos en tiempo real.
  - Utiliza **subsistemas OLAP** (a menudo en memoria) para manejar **grandes volúmenes** de datos rápidamente.



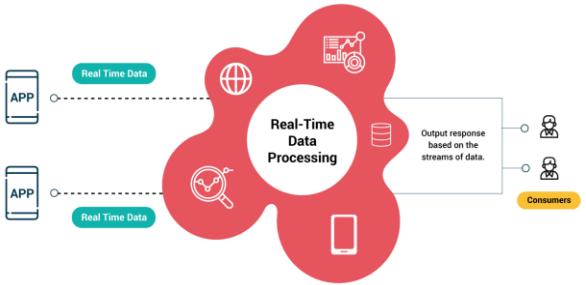
- **Aplicación:**

➢ Analíticas de tipo **descriptivo** e interactivas, donde es necesario consultar datos de forma **instantánea**.

4.3 Estrategias de procesamiento de datos

- **Procesamiento transaccional** es un tipo de **procesamiento en tiempo real**, ya que ambas requieren **respuestas inmediatas**.
- **Diferencia clave**:
  - El procesamiento **transaccional** siempre implica una **transacción**, mientras que el procesamiento **analítico en tiempo real** no.

➢ El término "tiempo real" puede aplicarse a ambos contextos, pero el **procesamiento transaccional** tiene un enfoque específico en la ejecución de transacciones, mientras que el **procesamiento analítico en tiempo real** se enfoca en **análisis y consultas** rápidas.

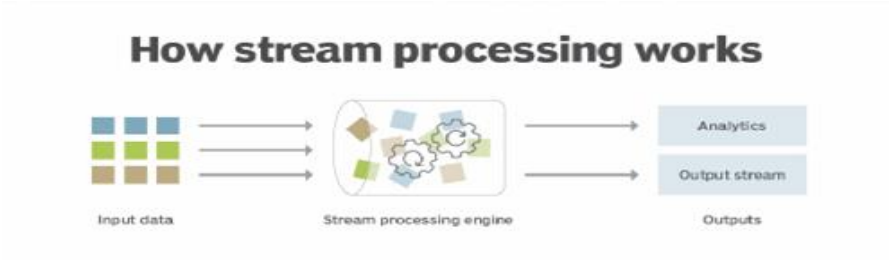


4.3 Estrategias de procesamiento de datos

- **Procesamiento en streaming:**
  - **Respuestas rápidas**, pero el procesamiento debe realizarse a la **misma velocidad** que se recibe el flujo de datos.
  - Se utiliza para manejar datos **continuos** que llegan en tiempo real.

- **Desafíos:**
  - Las estructuras de datos deben ser capaces de **actualizarse** en tiempo real a medida que llegan nuevos datos.
  - **Almacenamiento en memoria:** Impone un límite en el **tamaño** de los datos que pueden ser procesados simultáneamente.

➢ El procesamiento en **streaming** es clave para sistemas que manejan flujos de datos continuos, como transmisiones de redes sociales, sensores IoT o servicios de video en directo, pero presenta retos adicionales debido a la necesidad de **actualización continua** en tiempo real.



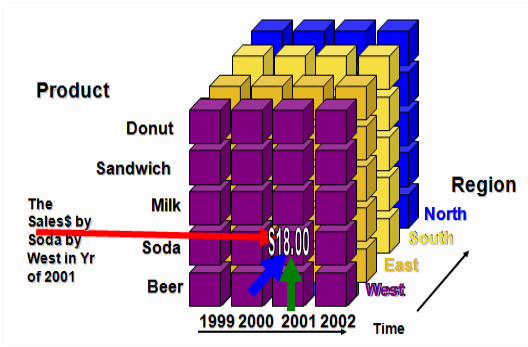
4.4 OLTP

- Orientado a **transacciones operacionales** que deben realizarse en **tiempo real**.
- Comúnmente utiliza **bases de datos relacionales**.

- **Acciones Simples**:
  - Las operaciones incluyen: **insertar**, **seleccionar**, **actualizar** o **eliminar** datos.
  - No incluye **analítica**, lo que permite tiempos de respuesta **inferiores a un segundo**.
- **Aplicación:**
  - Sistemas OLTP son usados en entornos donde las transacciones rápidas y **sin demoras** son esenciales para garantizar una **experiencia de usuario fluida**.

➢ El enfoque de **OLTP** es manejar operaciones transaccionales sencillas de manera **eficiente y rápida**, lo que lo hace ideal para aplicaciones de uso cotidiano donde la **velocidad** es crítica.

4.5 OLAP



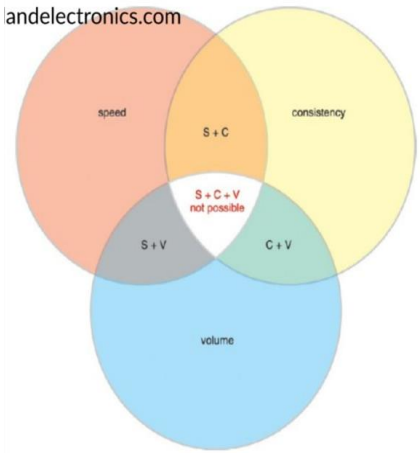
- **OLAP**:
  - Orientado a **análisis de datos** en tiempo real.
  - Usado en **todos los niveles** de la analítica de datos, desde informes simples hasta análisis complejos.

- **Bases de datos multidimensionales**:
  - Almacenan datos en estructuras llamadas **cubos OLAP**, optimizados para consultas complejas y rápidas.
  - Los datos están **desnormalizados** para mejorar el rendimiento.

- **Almacenamiento en memoria**:
  - A veces los datos se mantienen en **RAM**, lo que permite un acceso rápido, pero **limita** el tamaño de los datos que se pueden procesar simultáneamente.

➢ OLAP permite realizar **consultas complejas y rápidas** sobre grandes volúmenes de datos, siendo una herramienta clave en la toma de decisiones empresariales mediante la **analítica avanzada**.

4.6 Principio SCV



- **Principio SCV**:
  - Similar al teorema CAP, pero aplicado al **procesamiento de datos** en sistemas distribuidos.
  - Solo se pueden optimizar dos de sus características

- **Características del SCV**:
  - **Velocidad**: Procesamiento rápido de los datos.
  - **Consistencia**: Procesamiento consistente y preciso en todos los nodos.
  - **Volumen**: Capacidad de procesar grandes volúmenes de datos.

➢ En un sistema de procesamiento distribuido, el **principio SCV** requiere elegir entre **velocidad**, **consistencia**, o **volumen** de datos, sacrificando una de estas propiedades para optimizar las otras dos, lo que impacta en cómo se gestionan los recursos y las prioridades en sistemas distribuidos complejos.

4.6 Principio SCV

Velocidad:

- Se refiere a **cuánto tardan en procesarse los datos desde el momento en el que son recibidos en el sistema analítico**. Por lo general se excluye el tiempo que se tarda en capturar los datos, considerando sólo lo que se tarda en generar la estadística o ejecutar el algoritmo en cuestión.
- Esta velocidad es más alta si estamos ante un sistema de analítica en tiempo real que si se trata de un sistema de analítica por lotes (del inglés batch).

4.6 Principio SCV

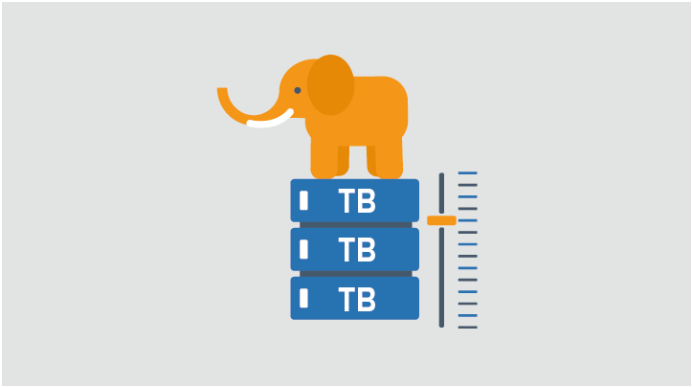
Consistencia:

- Se refiere en este caso a la **precisión de los resultados de la analítica** (no confundir, por lo tanto, con el significado de la C del teorema CAP).
- Tal precisión depende de si para la analítica se utilizan **todos los datos** disponibles (precisión **alta**) o de si por el contrario se emplean técnicas de **muestreo** para seleccionar sólo un **subconjunto** de los mismos con la intención de producir resultados (de **menor precisión**) en un menor tiempo.

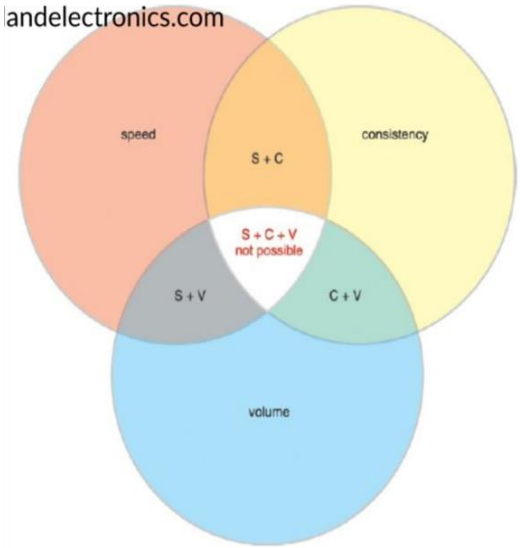
4.6 Principio SCV

Volumen:

- Se refiere a la **cantidad de datos que pueden ser procesados**.
- Hay que tener en cuenta que en entornos de Big Data, el alto volumen de datos es una características siempre presente (**una de las 5 Vs**).

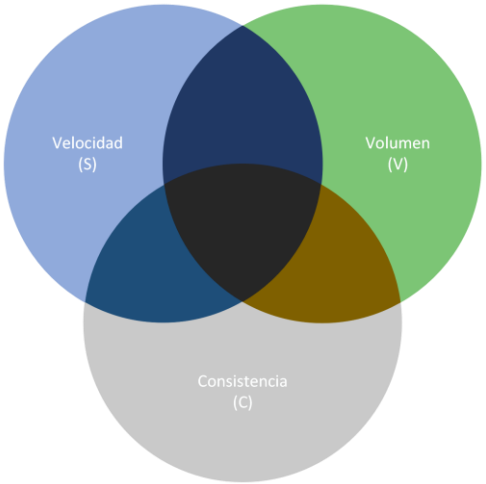


4.6 Principio SCV



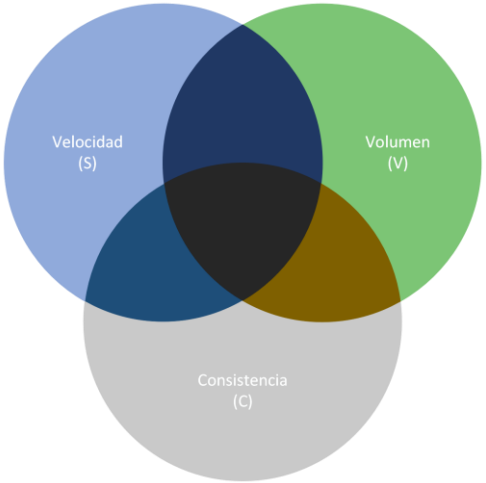
De igual modo que hicimos al estudiar el teorema CAP, nos fijaremos en una serie de escenarios para mostrar que no podemos conseguir un sistema que cumpla a la vez las 3 características del principio SCV.

4.6 Principio SCV – Escenarios- Velocidad y Consistencia (S + C)



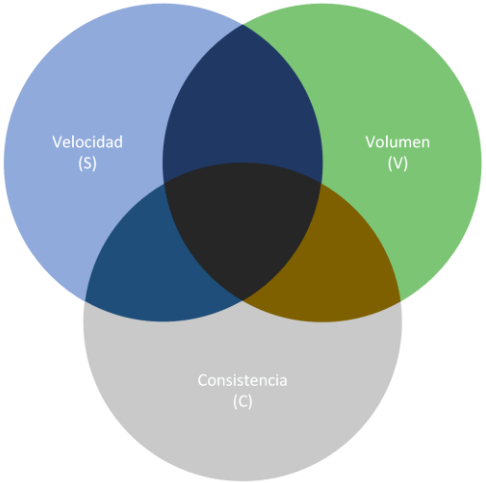
- S + C:
  - Se logra un procesamiento **rápido y consistente**, pero no se pueden manejar **grandes volúmenes de datos**.
  - **Limitación**: El sistema es incapaz de procesar datos en masa sin sacrificar velocidad o consistencia.

4.6 Principio SCV – Escenarios- Consistencia y Volumen (C + V)



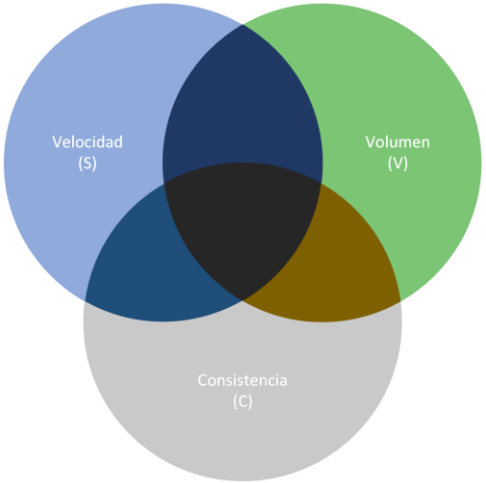
- C + V:
  - Se garantiza un procesamiento **preciso** sobre **grandes volúmenes de datos**, pero a costa de la **velocidad**.
  - **Limitación**: El procesamiento se vuelve más lento a medida que se asegura la consistencia en grandes volúmenes.

4.6 Principio SCV – Escenarios- Velocidad y Volumen (S + V)



- **S + V**:
  - Se logra procesar grandes volúmenes de datos a **alta velocidad**, pero sacrificando la **consistencia** mediante el uso de **muestreos** o subconjuntos de datos.
  - **Limitación**: Los resultados pueden no ser completamente precisos o consistentes.

4.6 Principio SCV – Escenarios- Conclusión



- El principio SCV muestra que es **imposible optimizar simultáneamente** la **velocidad**, **consistencia**, y **volumen** de datos en un sistema de procesamiento distribuido. Cada escenario requiere **sacrificar** una de las tres propiedades según las necesidades del sistema.

4.6 Principio SCV

Dado que en ambientes de Big Data el ser capaz de manejar grandes volúmenes de datos (V) es una obligación, ¿podremos típicamente **realizar analítica en tiempo real** con todos ellos?

No, ya que para que sea en tiempo real debemos cumplir S, por lo que necesitaríamos realizar la analítica sobre un subconjunto de los datos, encontrándonos en el caso S+V, y produciendo por lo tanto un resultado no totalmente consistente (C).

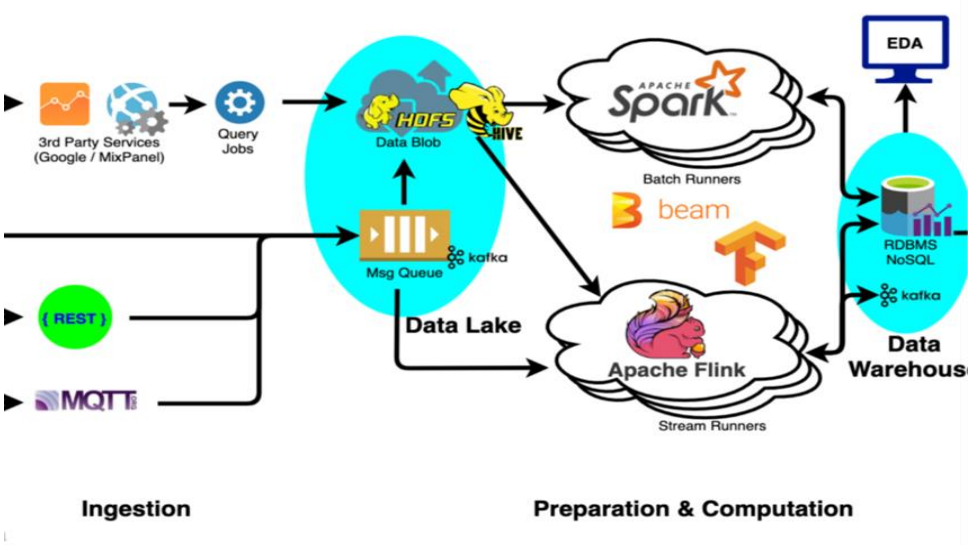
4.6 Principio SCV

Dado que en ambientes de Big Data el ser capaz de manejar grandes volúmenes de datos (V) es una obligación, ¿podremos típicamente realizar **procesamiento por lotes** empleando todo ese conjunto de datos?

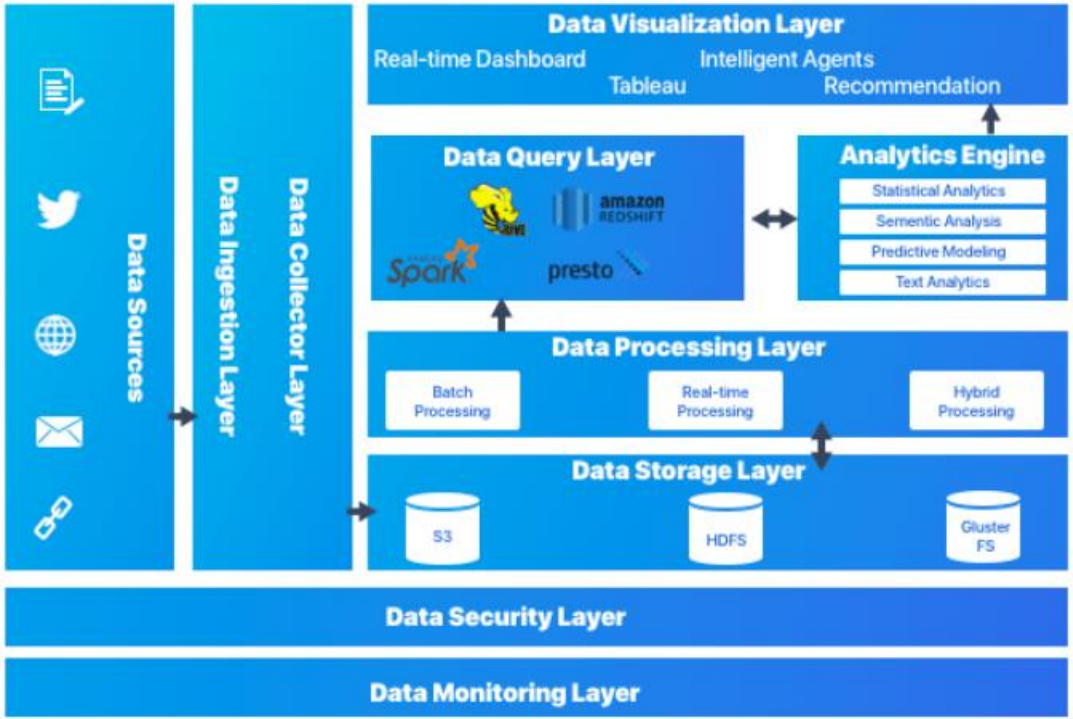
Sí, ya que al ser analítica por lotes en lugar de en tiempo real, la característica S ya no es necesaria, de modo que podemos encontrarnos en un caso C+V, utilizando todos los datos sin ningún tipo de muestreo para así producir un resultado consistente (C).

5 La arquitectura por capas de Big Data

Al margen de que durante el diseño y desarrollo de cada posible proyecto de Big Data pueda optarse por la estructura o arquitectura que más convenga, de modo generalizado se emplea una arquitectura según la cual el flujo de datos va pasando por una serie de capas.

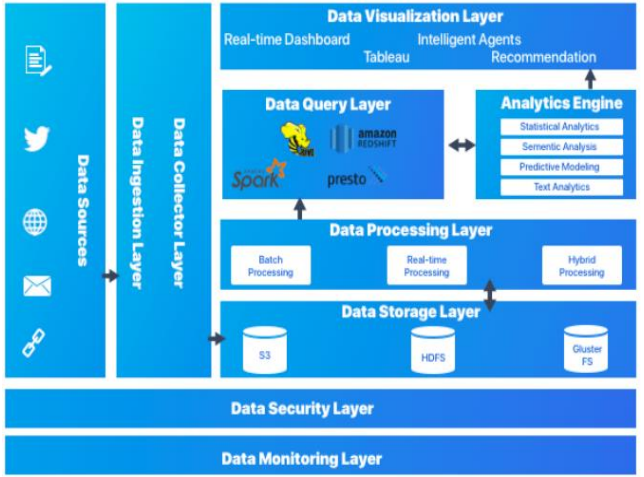


5 La arquitectura por capas de Big Data



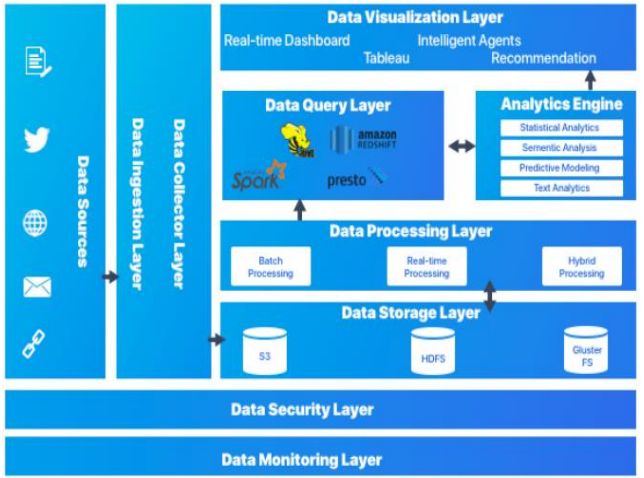
5 La arquitectura por capas de Big Data - Ingestión

- **Capa de Ingestión**:
  - Los datos provienen de **múltiples fuentes preexistentes**.
  - El sistema debe **adaptarse** a las fuentes, no al revés.
  - Usa **protocolos** adecuados para conectarse y **interpretar** los datos.



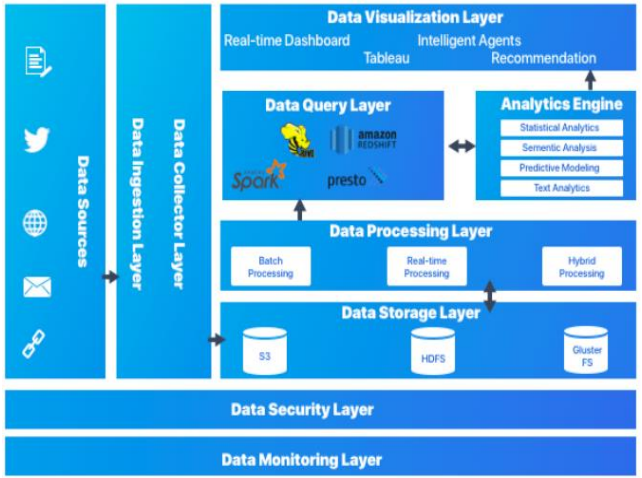
5 La arquitectura por capas de Big Data - Colección

- **Capa de Colección**:
  - **Integración** de datos provenientes de diversas fuentes.
  - Unificación de datos de **formatos diferentes** en una estructura coherente.
  - Los datos se preparan para ser **utilizados** de forma eficiente.



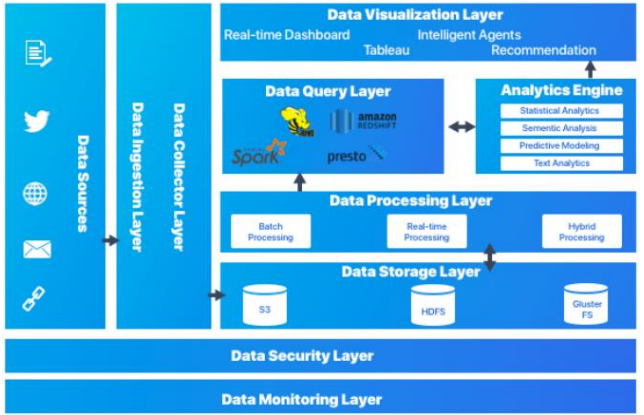
5 La arquitectura por capas de Big Data - Almacenamiento

- **Capa de Almacenamiento**:
  - Almacena grandes volúmenes de datos.
  - Utiliza sistemas de **almacenamiento distribuido** diseñados para **Big Data**.
  - Garantiza la **eficiencia y seguridad** en la gestión de datos masivos.



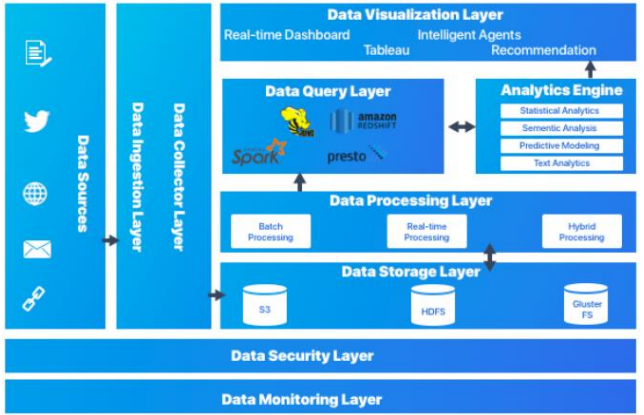
5 La arquitectura por capas de Big Data - Procesamiento

- **Capa de Procesamiento**:
  - **Procesa** datos a gran escala.
  - Soporta varios tipos de procesamiento: **lotes**, **tiempo real**, **streaming**, o **híbrido**.
  - Actúa según las instrucciones de la **capa de consulta**.



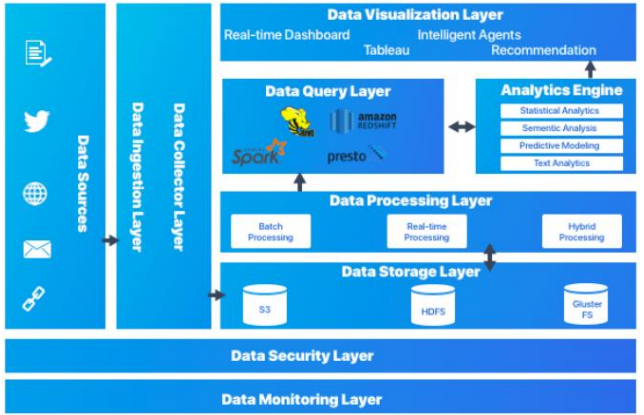
5 La arquitectura por capas de Big Data – Consulta y Analítica

- **Capa de Consulta y Analítica**:
  - **Obtiene valor** a partir de los datos procesados.
  - Realiza **análisis estadísticos** y **algoritmos** para generar insights.
  - Dependiente de los datos procesados en la **capa de procesamiento**.



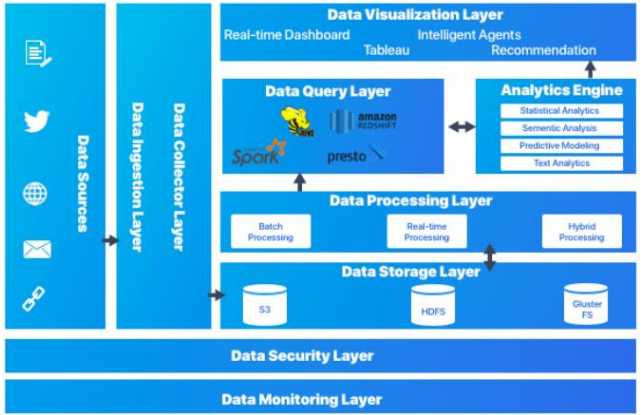
5 La arquitectura por capas de Big Data - Visualización

- **Capa de Visualización**:
  - Permite **consultar informes** y acceder a **cuadros de mando interactivos**.
  - Proporciona opciones para visualizar y manipular los datos.
  - Es la capa desde la que se toman **decisiones de negocio**.



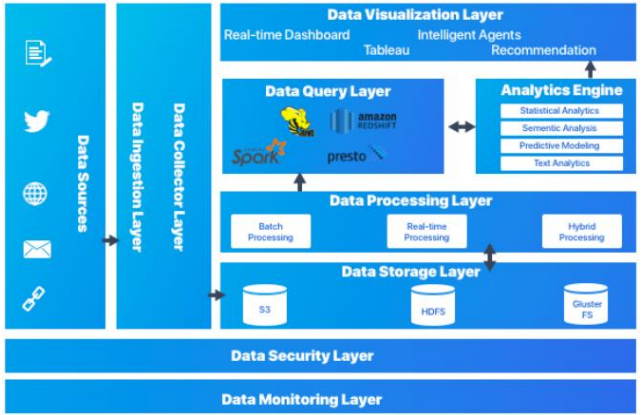
5 La arquitectura por capas de Big Data - Seguridad

- **Capa de Seguridad**:
  - Asegura los datos ante ataques y usos malintencionados.
  - Utiliza **métodos físicos** (protección de hardware) y **de software** (cifrado, autenticación).
  - Protege tanto de amenazas **internas** como **externas**.



5 La arquitectura por capas de Big Data - Monitorización

- **Capa de Monitorización**:
  - Supervisa el **estado** de los datos y del sistema.
  - Incluye **auditoría**, **testeo**, **gestión**, y **control** de los datos.
  - Asegura que los datos sean **precisos**, **actualizados** y adecuados para análisis.



6 El paisaje de Big Data o Big Data Landscape

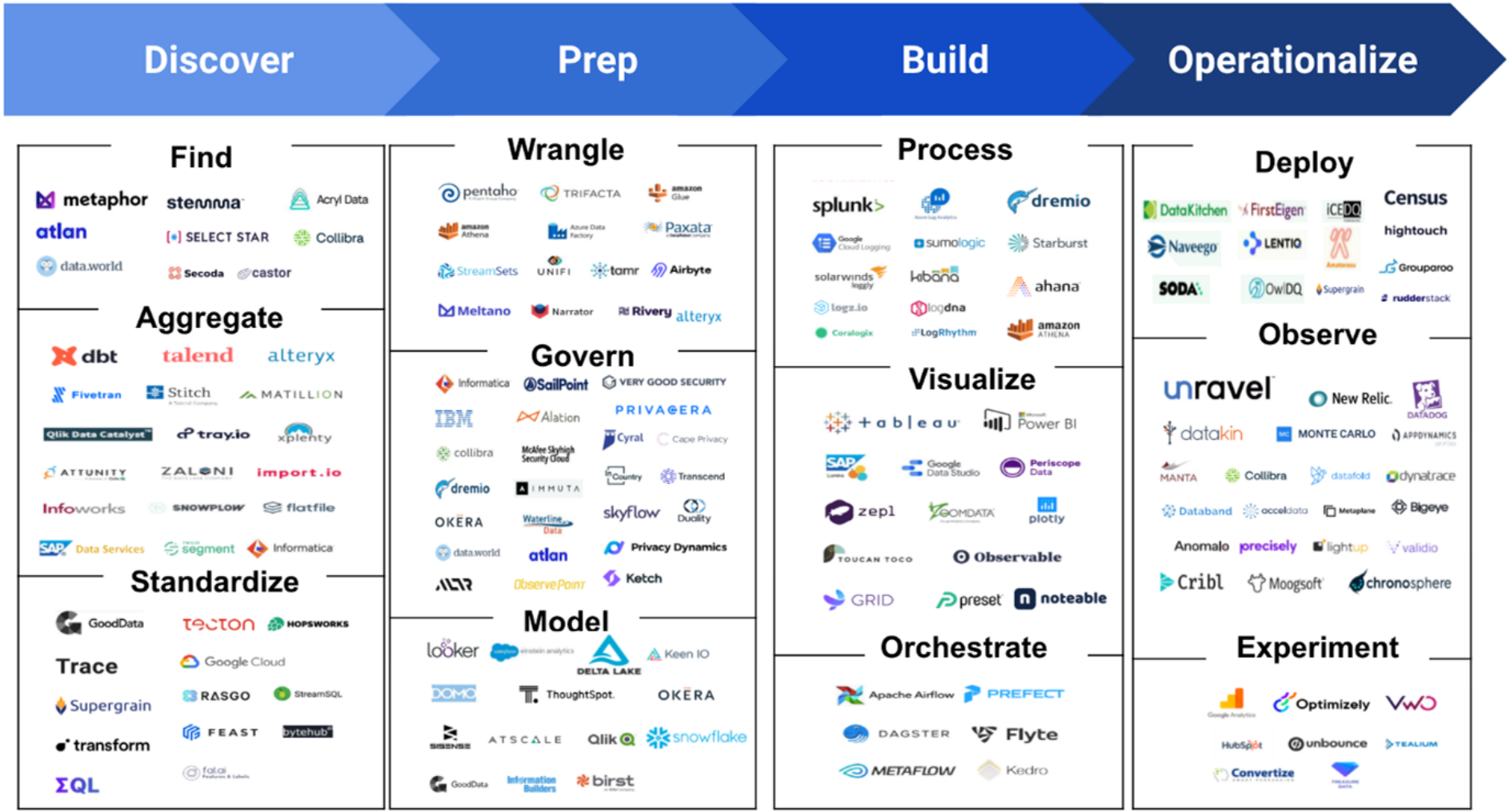
- **Hadoop:**
  - Plataforma pionera para Big Data, especializada en procesamiento por lotes.
  - Ecosistema Hadoop:
    - Herramientas diseñadas para cada fase del trabajo con datos, desde la ingestión hasta el almacenamiento y procesamiento.

- **Apache Spark:**
  - Plataforma enfocada en procesamiento en tiempo real y streaming.
  - Interacción con el ecosistema Hadoop, considerada por algunos como parte del mismo.

- **Bases de datos NoSQL y NewSQL:**
  - Soluciones especializadas en almacenamiento para casos y necesidades específicas.

- **Herramientas de Analítica y Visualización:**
  - Gran variedad de herramientas subclasificadas por función dentro del análisis y visualización de datos.

- **Aplicaciones específicas:**
  - Utilizan o interactúan con las herramientas mencionadas, integrándose en el panorama de Big Data.


Discover
Prep
Build
Operationalize
Find
metaphor
stemma
Acryl Data
atlan
SELECT STAR
Collibra
data.world
Secoda
castor
Aggregate
dbt
talend
alteryx
Fivetran
Stitch
MATILLION
Qlik Data Catalyst
tray.io
xplenty
ATTUNITY
ZALONI
import.io
Infoworks
SNOWPLOW
flatfile
Data Services
segment
Informatica
Standardize
GoodData
tecton
HOPSWORKS
Trace
Google Cloud
Supergrain
RASGO
StreamSQL
transform
FEAST
bytehub
SQL
Wrangle
pentaho
TRIFACTA
amazon Glue
amazon Athena
Azure Data Factory
Paxata
StreamSets
UNIFI
tamr
Airbyte
Meltano
Narrator
Rivery
alteryx
Govern
Informatica
SailPoint
VERY GOOD SECURITY
IBM
Alation
PRIVACERA
collibra
McAfee Skyhigh Security Cloud
Cyral
Cape Privacy
dremio
IMMUTA
Country
Transcend
OKERA
Waterline Data
skyflow
Duality
data.world
atlan
Privacy Dynamics
ALTR
ObservePoint
Ketch
Model
looker
einstein analytics
DELTA LAKE
Keen IO
DOMO
ThoughtSpot
OKERA
SISENSE
ATSCALE
Qlik
snowflake
GoodData
Information Builders
birst
Process
splunk>
dremio
Google Cloud Logging
sumologic
Starburst
solarwinds loggly
kibana
ahana
logz.io
logdna
Coralogix
LogRhythm
amazon ATHENA
Visualize
tableau
Power BI
SAP
Google Data Studio
Periscope Data
zepl
GOODATA
plotly
TOUCAN TOCO
Observable
GRID
preset
noteable
Orchestrate
Apache Airflow
PREFECT
DAGSTER
Flyte
METAFLOW
Kedro
Deploy
DataKitchen
FirstEigen
iCEDQ
Census
hightouch
Naveego
LENTIQ
Amatrasu
Grouparoo
SODA
OwlDQ
Supergrain
rudderstack
Observe
unravel
New Relic.
DATADOG
datakin
MONTE CARLO
APPDYNAMICS
MANTA
Collibra
datafold
dynatrace
Databand
acceldata
Metaplane
Bigeye
Anomalo
precisely
lightup
validio
Cribl
Moogsoft
chronosphere
Experiment
Google Analytics
Optimizely
VWO
HubSpot
unbounce
TEALIUM
Convertize


THE
END