1. ¿Por qué Big Data?

- Big Data surge como respuesta al desafío de procesar volúmenes de información que los sistemas tradicionales no pueden manejar en una sola máquina.
- Su enfoque actual se centra en el análisis del comportamiento de usuarios y en la extracción de valor a través de modelos predictivos y patrones de datos.
- Combina almacenamiento **distribuido** y **procesamiento paralelo** para escalar horizontalmente.
- Se estima que el volumen mundial de datos alcanzará los **163 zettabytes en 2025**.

1.1 Las 5 Vs de Big Data

1. **Volumen:** Cantidad masiva de datos generados por transacciones, sensores, IoT, redes sociales, etc.
2. **Velocidad:** Ritmo con el que los datos se generan y deben procesarse. Da lugar al procesamiento **en tiempo real** y **en streaming**.
3. **Variedad:** Diferentes tipos y formatos:
    - Estructurados (bases de datos relacionales)
    - Semiestructurados (CSV, JSON, XML)
    - No estructurados (videos, audios, imágenes)
4. **Veracidad:** Grado de confianza de los datos; depende de su origen y calidad.
5. **Valor:** Utilidad que aportan a la toma de decisiones. Aumenta si los datos son veraces y se procesan con rapidez.

1.2 Beneficios

- Almacenamiento **distribuido y replicado**, que garantiza disponibilidad y tolerancia a fallos.
- Permite **optimizar operaciones**, **predecir comportamientos** y **tomar decisiones basadas en evidencia**.
- Ciclo de valor del dato: **Eventos → Datos → Información → Conocimiento → Sabiduría → Valor.**

2. Clústeres de Computadoras

- Un **clúster** es un conjunto de nodos conectados que funcionan como una unidad lógica.
- Se construye con **hardware común (commodity hardware)** dentro de un **CPD** (Centro de Procesamiento de Datos).
- **Ventajas:**
    - **Rendimiento:** Procesamiento paralelo.
    - **Disponibilidad:** Tolerancia a fallos.
    - **Escalabilidad horizontal:** Se amplía añadiendo más nodos (scale-out).
- **Comunicaciones:**
    - Más rápidas dentro del CPD (por switches de alta velocidad).
    - Más lentas hacia sistemas externos.

3. Almacenamiento y Principios Fundamentales

3.1 Tipos

- **Bases de datos relacionales (RDBMS):** Escalables verticalmente; adecuadas para transacciones.
- **Data Warehouse:** Almacén estructurado de datos históricos y actuales para analítica.
- **Data Lake:** Almacenamiento masivo de datos en bruto, de todo tipo y formato.

3.2 Propiedades ACID

- **Atomicidad:** La transacción se completa totalmente o se revierte.
- **Consistencia:** Los datos cumplen las reglas del sistema tras cada transacción.
- **Aislamiento:** Una transacción no afecta a otra hasta completarse.
- **Durabilidad:** Los cambios son permanentes, aunque haya fallos.

3.3 Teorema CAP

Una base de datos distribuida solo puede garantizar dos de tres:

- **Consistencia (C)**
- **Disponibilidad (A)**
- **Tolerancia a particiones (P)**

En Big Data se suele priorizar **P + A**, sacrificando la consistencia inmediata.

3.4 Modelo BASE

- **Básicamente disponible**, **estado blando**, **consistencia eventual**.
- Favorece la disponibilidad y la escalabilidad.
- No apto para sistemas transaccionales, pero ideal para análisis masivo.

4. Procesamiento de Datos

4.1 Procesamiento Distribuido y Paralelo

- Los datos se procesan en distintos nodos de un clúster, aprovechando varios niveles de paralelismo.
- La comunicación entre nodos se optimiza para reducir los saltos entre switches.

4.2 Estrategias de Procesamiento

- **Por lotes (Batch):** Grandes volúmenes, resultados diferidos.
- **Transaccional (OLTP):** Operaciones inmediatas y seguras.
- **En tiempo real (OLAP):** Analítica instantánea y consultas rápidas.
- **Streaming:** Procesamiento continuo a la velocidad de llegada del dato, ideal para sensores, IoT o redes sociales.

4.3 Principio SCV

- No se pueden optimizar simultáneamente **Velocidad, Consistencia y Volumen**.
- Escenarios:
    - **C + V:** Procesamiento por lotes (más preciso, más lento).
    - **S + V:** Analítica en tiempo real (más rápido, menos preciso).

5. Arquitectura por Capas de Big Data

1. **Ingestión:** Captura datos de múltiples fuentes.
2. **Colección:** Integra y unifica datos de distintos formatos.
3. **Almacenamiento:** Guarda datos en sistemas distribuidos (Data Lake).
4. **Procesamiento:** Transforma datos (batch, real time o streaming).
5. **Consulta y Analítica:** Obtiene valor mediante algoritmos o modelos.
6. **Visualización:** Cuadros de mando e informes para la toma de decisiones.
7. **Seguridad y Monitorización:** Controlan accesos, auditorías y rendimiento.

6. Paisaje de Big Data

- **Hadoop:** Plataforma pionera; se centra en el procesamiento por lotes en disco (HDFS, MapReduce, YARN, Hive).
- **Apache Spark:** Procesamiento en memoria para tareas en tiempo real y streaming.
- **Bases de datos NoSQL:** Tipos clave: clave-valor, columnares, documentales y gráficas.
- **NewSQL:** Bases transaccionales modernas que combinan escalabilidad y consistencia.
- **Herramientas de analítica y visualización:** Power BI, Tableau, Qlik, entre otras.

COMANDOS

I. Fundamentos

- `pip install pandas` / `conda install pandas`: Instala la librería Pandas.
- `import pandas as pd`: Importa Pandas con el alias estándar.
- `pd.Series()`: Crea una serie unidimensional.
- `pd.DataFrame()`: Crea una tabla bidimensional (DataFrame).

II. Entrada/Salida (I/O)

- `pd.read_csv('archivo.csv')`: Lee datos desde un archivo CSV o URL.
- `pd.read_excel('archivo.xlsx')`: Lee datos desde un archivo Excel.
- `pd.read_json('archivo.json')`: Lee datos desde un archivo o API JSON.
- `df.to_csv('salida.csv', index=False)`: Guarda el DataFrame como CSV sin índice.
- `pd.DataFrame(datos)`: Crea un DataFrame desde listas o diccionarios.
- `parse_dates=['columna']`: Convierte una columna a formato fecha al leer el CSV.

III. Exploración y Limpieza

- `df.head(n)`: Muestra las primeras *n* filas.
- `df.tail(n)`: Muestra las últimas *n* filas.
- `df.shape`: Devuelve (filas, columnas) del DataFrame.
- `df.columns`: Lista los nombres de las columnas.
- `df.info()`: Muestra estructura y tipos de datos.
- `df.describe()`: Calcula estadísticas básicas de las columnas numéricas.
- `df.isna().sum()`: Cuenta los valores nulos por columna.
- `df.dropna(subset=['columna'])`: Elimina filas con valores nulos en una columna.
- `df['col'].fillna(valor)`: Rellena valores nulos con un valor dado.
- `df.drop_duplicates()`: Elimina filas duplicadas.
- `df.rename(columns={'old':'new'})`: Renombra columnas.

IV. Manipulación y Análisis

- `df['columna']` o `df.columna`: Accede a una columna.
- `df[['col1','col2']]`: Accede a varias columnas.
- `df.loc[fila, 'col']`: Selecciona por etiqueta.
- `df.iloc[fila, col]`: Selecciona por posición.
- `df[df['col'] > x]`: Filtra filas según una condición.
  `(cond1) & (cond2)`: Combina condiciones booleanas (AND lógico).
- `df['nueva'] = df['col1'] + df['col2']`: Crea o modifica columnas.
- `df.sort_values('columna')`: Ordena el DataFrame por una o más columnas.
- `df['columna'].value_counts()`: Cuenta valores únicos en una columna.
- `df['columna'].sum()`: Calcula la suma de una columna.
- `df.mean()`: Calcula la media de columnas numéricas.
- `df.groupby('col')['otra'].sum()`: Agrupa por una columna y agrega datos.
- `pd.pivot_table(df, values, index, columns, aggfunc)`: Crea una tabla pivote.

V. Visualización

- `import matplotlib.pyplot as plt`: Importa la librería de gráficos base.
- `df['columna'].value_counts().plot.bar()`: Gráfico de barras.
- `df.boxplot(column=['col'], by='otra')`: Boxplot por grupo.
- `df[['col1', 'col2']].plot.line()`: Gráfico de líneas de varias columnas.