SESIÓN BIG DATAL

CONTENIDOS:

- Qué es big data.
- Las 5v's de big data.
- ¿qué problemas se pueden resolver con big data?
- Sistemas locales y distribuidos.
- Tecnologías de bigdata
  - NoSQL
  - Real Time
  - Almacenamiento distribuido
  - Procesamiento distribuido
  - Frameworks
- Arquitecturas típicas de big data.
- Evolución de las tecnologías de bigdata
  - Hadoop
  - Spark
  - Otras

QUÉ ES BIG DATA

Big Data se refiere al manejo y análisis de grandes volúmenes de datos que son demasiado complejos o grandes para ser procesados con herramientas tradicionales. No se trata solo del tamaño de los datos, sino también de la velocidad a la que se generan y la variedad de formatos en los que pueden estar (texto, imágenes, videos, sensores, etc.).

Se usa en múltiples industrias, desde la salud hasta el comercio, para extraer información valiosa y mejorar la toma de decisiones.

LAS 5V'S DE BIG DATA

Para entender el alcance de Big Data, se utilizan las 5V's:

1. **Volumen**: Cantidad masiva de datos que se generan constantemente.

2. **Velocidad**: Ritmo al que los datos son creados y procesados en tiempo real o casi real.
3. **Variedad**: Diferentes formatos de datos (estructurados, no estructurados, semiestructurados).
4. **Veracidad**: Calidad y confiabilidad de los datos para tomar decisiones correctas.
5. **Valor**: Información útil que se puede extraer para generar conocimiento o impacto en un negocio.

Ilustración 1 Imagen ilustrativa Big Data Roberto Jasinki

¿QUÉ PROBLEMAS SE PUEDEN RESOLVER CON BIG DATA?

Big Data ayuda a resolver problemas en distintos sectores:

- **Salud**: Análisis de datos clínicos para diagnósticos más precisos.
- **Finanzas**: Detección de fraudes en transacciones bancarias.
- **Marketing**: Personalización de publicidad según patrones de comportamiento.
- **Transporte**: Optimización de rutas y predicción de fallas en vehículos.
- **Ciencia**: Análisis de grandes volúmenes de datos genéticos o climáticos.

SISTEMAS LOCALES Y DISTRIBUIDOS

1. **Sistemas Locales**: Procesan datos en un solo servidor o computadora. Son útiles para tareas pequeñas, pero tienen límites en almacenamiento y procesamiento.
2. **Sistemas Distribuidos**: Dividen la carga de trabajo en múltiples servidores para procesar datos de manera eficiente. Son la base de tecnologías como Hadoop y Spark.

TECNOLOGÍAS DE BIGDATA

Big Data requiere herramientas y enfoques específicos para almacenar, procesar y analizar grandes volúmenes de datos de manera eficiente. Entre las principales tecnologías utilizadas encontramos:

1. **NoSQL**

   Los sistemas de bases de datos tradicionales (SQL) no siempre son eficientes para manejar los grandes volúmenes y variedad de datos de Big Data. Por eso, surgieron las bases de datos **NoSQL (Not Only SQL)**, que ofrecen flexibilidad y escalabilidad.

   Características clave de NoSQL:

   - **Escalabilidad horizontal**: Se pueden distribuir los datos en múltiples servidores sin afectar el rendimiento.
   - **Flexibilidad en los esquemas**: No requieren una estructura fija como las bases de datos relacionales.
   - **Soporte para datos semiestructurados y no estructurados** (JSON, XML, documentos, gráficos, etc.).

   Tipos de bases de datos NoSQL:

   - **Clave-valor (Key-Value Stores)**: Datos almacenados en pares clave-valor. Ej: Redis, DynamoDB.
   - **Columnar (Wide-Column Stores)**: Organizan datos en columnas en lugar de filas. Ej: Apache Cassandra, HBase.
   - **Documentales (Document Stores)**: Almacenan documentos en formato JSON o BSON. Ej: MongoDB, CouchDB.
   - **Grafos (Graph Databases)**: Útiles para representar relaciones complejas. Ej: Neo4j, ArangoDB.

Ilustración 2 Bases de datos NoSQL. Tecnopedia

2. **Real Time (Procesamiento en tiempo real)**

   El procesamiento en tiempo real se refiere a la capacidad de analizar y actuar sobre los datos a medida que se generan, sin esperar a que se acumulen grandes volúmenes.

   Ejemplos de aplicaciones en tiempo real:

   - **Análisis de fraudes financieros**: Detección de transacciones sospechosas en bancos.
   - **Monitoreo de redes sociales**: Análisis de tendencias y sentimientos en Twitter, Facebook, etc.
   - **Sistemas de recomendación**: Sugerencias en Netflix, YouTube o Spotify en función del comportamiento del usuario.
   - **Monitoreo de sensores IoT**: Diagnóstico en tiempo real de fallas en maquinaria industrial o automóviles conectados.

   Algunas tecnologías utilizadas en procesamiento en tiempo real:

   - **Apache Kafka**: Plataforma de mensajería para transmisión de datos en tiempo real.
   - **Apache Flink**: Procesamiento distribuido de datos en flujo continuo.
   - **Apache Storm**: Procesamiento de eventos en tiempo real.
   - **Spark Streaming**: Extensión de Apache Spark para trabajar con datos en flujo.

Ilustración 3 Logo Apache Kafka

3. **Almacenamiento distribuido**

   Cuando los datos son demasiado grandes para almacenarse en un solo servidor, se utilizan sistemas de almacenamiento distribuido, que reparten los datos en múltiples nodos o servidores.

Ejemplos de sistemas de almacenamiento distribuido:

- **HDFS (Hadoop Distributed File System)**: Sistema de archivos distribuido utilizado en el ecosistema Hadoop.
- **Amazon S3**: Solución de almacenamiento en la nube escalable de AWS.
- **Google Cloud Storage**: Servicio de almacenamiento distribuido de Google.
- **Ceph**: Sistema de almacenamiento distribuido open-source.

Características clave del almacenamiento distribuido:

- **Replicación**: Los datos se almacenan en múltiples ubicaciones para evitar pérdida de información.
- **Tolerancia a fallos**: Si un nodo falla, los datos pueden recuperarse de otros nodos.
- **Escalabilidad**: Se pueden agregar más nodos para aumentar la capacidad de almacenamiento.

Ilustración 4 Logo Amazon S3

4. **Procesamiento distribuido**

   Para analizar grandes volúmenes de datos de manera eficiente, se utilizan técnicas de procesamiento distribuido, en las que múltiples servidores trabajan en paralelo para dividir la carga de trabajo.

   Principales enfoques:

   - **Batch Processing (Procesamiento por lotes)**: Se procesa un gran volumen de datos en intervalos de tiempo. Ej: Hadoop MapReduce.

- **Stream Processing (Procesamiento en flujo)**: Se procesan los datos en tiempo real a medida que llegan. Ej: Apache Flink, Apache Storm.

Ejemplos de tecnologías de procesamiento distribuido:

- **Apache Hadoop**: Framework basado en el modelo MapReduce para el procesamiento de grandes volúmenes de datos.
- **Apache Spark**: Plataforma de procesamiento distribuido en memoria, más rápida que Hadoop.
- **Google BigQuery**: Servicio en la nube para análisis de grandes volúmenes de datos.

Ilustración 5 Logo Apache Spark

5. **Frameworks**

Los frameworks proporcionan herramientas y bibliotecas para facilitar el almacenamiento, procesamiento y análisis de datos a gran escala.

Algunos de los frameworks más importantes en el ecosistema de Big Data:

1. Apache Hadoop

- Basado en el modelo MapReduce.
- Almacena datos en HDFS (Hadoop Distributed File System).
- Procesamiento de grandes volúmenes de datos en clústeres de computadoras.

2. Apache Spark

- Procesamiento en memoria, más rápido que Hadoop.
- Soporta Batch Processing y Stream Processing.
- Compatible con lenguajes como Python, Java y Scala.

3. Apache Flink

- Framework para procesamiento en tiempo real.
- Ofrece baja latencia y alto rendimiento.

4. Apache Storm

- Framework especializado en procesamiento de eventos en tiempo real.
- Se usa en análisis de logs, detección de fraudes, IoT, etc.

5. Dask

- Framework en Python para procesamiento paralelo de datos.
- Compatible con Pandas y NumPy.

Ilustración 6 Logo Dask

ARQUITECTURAS TÍPICAS DE BIG DATA

Las arquitecturas de Big Data son modelos que organizan cómo se recopilan, almacenan, procesan y analizan grandes volúmenes de datos de manera eficiente. Existen varios enfoques según los requerimientos del sistema, pero en general, siguen una estructura basada en capas.

Arquitectura en Capas de Big Data

Una arquitectura típica de Big Data se divide en varias capas, cada una con funciones específicas:

1. Capa de Ingesta de Datos

Se encarga de la recopilación y entrada de datos desde múltiples fuentes. Puede incluir datos estructurados, semiestructurados y no estructurados provenientes de:

- Bases de datos tradicionales (SQL, NoSQL).
- Sensores IoT.
- Redes sociales.
- Logs de servidores.
- APIs externas.

Ejemplos de herramientas:

- **Apache Kafka**: Para transmisión de datos en tiempo real.
- **Flume**: Para ingesta de logs y eventos.
- **Sqoop**: Para importar/exportar datos entre Hadoop y bases de datos relacionales.

2. Capa de Almacenamiento

Los datos deben ser almacenados de manera escalable y tolerante a fallos. Existen diferentes soluciones según el tipo de datos:

- **Datos estructurados** → Bases de datos NoSQL como MongoDB, Cassandra.
- **Datos en grandes volúmenes** → Sistemas de archivos distribuidos como HDFS.
- **Datos en la nube** → Amazon S3, Google Cloud Storage.

Ejemplos de tecnologías:

- **HDFS (Hadoop Distributed File System)**
- **Amazon S3, Google Cloud Storage, Azure Blob Storage**
- **Apache Cassandra, MongoDB** (bases NoSQL para almacenamiento distribuido)

3. Capa de Procesamiento

Aquí es donde los datos se transforman y analizan. Puede haber dos tipos de procesamiento:

1. **Procesamiento por lotes (Batch Processing):** Se ejecutan tareas en grandes volúmenes de datos almacenados.
    - **Ejemplo:** Hadoop MapReduce, Apache Spark.
2. **Procesamiento en tiempo real (Stream Processing):** Se analizan los datos a medida que llegan.
    - **Ejemplo:** Apache Flink, Apache Storm, Spark Streaming.

4. Capa de Análisis y Machine Learning

Después del procesamiento, se realizan análisis avanzados o modelos de aprendizaje automático.

Ejemplos de herramientas:

- **Apache Mahout**: Machine Learning escalable en Hadoop.
- **MLlib**: Biblioteca de Machine Learning de Spark.
- **TensorFlow, PyTorch**: Modelos avanzados de IA.

5. Capa de Visualización e Interfaz de Usuario

Los resultados del análisis deben ser presentados de manera comprensible para la toma de decisiones.

Ejemplos de herramientas:

- **Tableau, Power BI**: Dashboards y análisis visual.
- **Grafana**: Monitoreo y visualización de métricas.
- **Jupyter Notebooks**: Análisis interactivo en Python.

Modelos de Arquitectura en Big Data

Dependiendo del uso y necesidades, existen varias arquitecturas comunes:

1. Arquitectura Lambda (Batch + Streaming)

Esta arquitectura combina **procesamiento en lotes** y **procesamiento en tiempo real** para manejar grandes volúmenes de datos de manera eficiente.

Componentes:

- **Capa Batch:** Procesamiento masivo con Hadoop/Spark.
- **Capa Streaming:** Procesamiento en tiempo real con Apache Flink o Spark Streaming.
- **Capa de Servición:** Almacena y visualiza los resultados para consultas en tiempo real.

Usos:

- Análisis de logs en servidores.
- Recomendaciones en e-commerce.

2. Arquitectura Kappa (Solo Streaming)

Mejora la arquitectura Lambda eliminando la capa de Batch y procesando todos los datos en tiempo real.

Ventajas:

- Más simple que Lambda.
- Bajo tiempo de latencia.

Ejemplo de uso:

- Análisis de datos de sensores IoT en tiempo real.

3. Arquitectura en Data Lake

Un **Data Lake** almacena datos sin procesar en su formato original y permite analizarlos posteriormente según sea necesario.

Componentes:

- Almacenamiento en HDFS o Amazon S3.
- Procesamiento con Spark o Presto.

Ejemplo de uso:

- Empresas que necesitan guardar todos sus datos sin estructurar para futuras consultas.

EVOLUCIÓN DE LAS TECNOLOGÍAS DE BIGDATA

El ecosistema de Big Data ha evolucionado con el tiempo para mejorar la eficiencia en el procesamiento y almacenamiento de grandes volúmenes de datos. Esta evolución ha pasado por diferentes etapas, desde los primeros sistemas de procesamiento distribuido hasta soluciones más avanzadas y optimizadas como Apache Spark.

Hadoop: El inicio del procesamiento distribuido

Apache Hadoop fue una de las primeras tecnologías en permitir el almacenamiento y procesamiento de grandes volúmenes de datos de manera distribuida.

Características principales:

- ✓ **HDFS (Hadoop Distributed File System):** Sistema de archivos distribuido que permite almacenar datos en múltiples nodos.
- ✓ **MapReduce:** Modelo de programación para procesar datos en paralelo en diferentes nodos.
- ✓ **YARN (Yet Another Resource Negotiator):** Administrador de recursos para coordinar el uso de nodos en el clúster.

Ventajas y Desventajas

Ventajas	Desventajas
Escalabilidad horizontal: Puede crecer agregando más máquinas al clúster.	**Alta latencia:** No es eficiente para análisis en tiempo real, ya que sigue un esquema Batch.
Tolerancia a fallos: Si un nodo falla, los datos se replican en otros nodos.	**Procesamiento lento:** Depende de la escritura/lectura en disco, lo que limita la velocidad.

Spark: El siguiente paso en el procesamiento distribuido

Apache Spark surgió como una alternativa más rápida y eficiente a Hadoop, al aprovechar la **memoria RAM** en lugar de depender solo del almacenamiento en disco.

Características principales:

- ✓ **Procesamiento en memoria:** Spark almacena datos en RAM, lo que mejora significativamente la velocidad.
- ✓ **Compatibilidad con Hadoop:** Puede ejecutarse sobre HDFS.
- ✓ **Múltiples módulos integrados:**
  - **Spark SQL**: Consultas estructuradas sobre Big Data.
  - **MLlib**: Biblioteca de Machine Learning.
  - **GraphX**: Procesamiento de grafos.
  - **Spark Streaming**: Procesamiento de datos en tiempo real.

Ventajas y Desventajas

Ventajas	Desventajas
Velocidad: Hasta 100 veces más rápido que Hadoop en ciertas tareas.	**Uso intensivo de memoria:** Puede consumir mucha RAM, lo que requiere hardware potente.
Procesamiento en tiempo real: Compatible con flujos de datos en vivo.	**Curva de aprendizaje:** Más complejo que Hadoop.
Soporte para múltiples lenguajes: Python, Scala, Java, R.	

Otras Tecnologías de Big Data

A medida que el ecosistema de Big Data ha evolucionado, han surgido tecnologías especializadas para cubrir necesidades específicas. Antes vimos ya estas tecnologías que son: Frameworks modernos, Almacenamiento en la nube, Procesamiento en tiempo real y bases de datos NoSQL.