Fundamentos de

Big Data

Sesión 1

¿Qué es Big Data?

☞ **Definición**
Manejo y análisis de grandes volúmenes de datos.

☞ **Complejidad**
Datos complejos para herramientas tradicionales.

☞ **Características**
Velocidad y variedad de formatos.

☞ **Aplicaciones**
Salud, comercio y más industrias.

Las 5v’s de Big Data

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.

Problemas resueltos con Big Data

Salud

Diagnósticos precisos con datos clínicos

Finanzas

Detección de fraudes bancarios.

Marketing

Publicidad personalizada.

Transporte

Optimización de rutas.

Imagen ilustrativa Big Data Roberto Jasinki

Sistemas: Locales vs Distribuidos

Locales:
Procesan datos en un solo servidor.
Limitaciones: Capacidad limitada de almacenamiento y procesamiento.

Distribuidos:
Dividen la carga entre múltiples servidores, permitiendo escalar horizontalmente.
Base tecnológica: Hadoop y Spark.

Componentes principales de Hadoop

- **HDFS (Hadoop Distributed File System):**
  Sistema de archivos distribuido que almacena grandes volúmenes de datos en bloques repartidos entre nodos.
- **YARN (Yet Another Resource Negotiator):**
  Gestor de recursos que asigna y coordina el uso de CPU y memoria en el clúster.
- **MapReduce:**
  Modelo de programación que divide tareas en dos fases (Map y Reduce) para el procesamiento paralelo de datos.

Componentes principales de Spark

- **Driver:**
  Programa principal que coordina la ejecución, gestiona el flujo de trabajo y la planificación de tareas.
- **Executor:**
  Procesos que ejecutan las tareas asignadas por el driver y almacenan resultados intermedios.
- **Cluster Manager:**
  Coordina los recursos del clúster (Spark puede usar YARN, Mesos o su propio gestor).

Tecnologías de Big Data: NoSQL

Escalabilidad
Distribución en múltiples servidores.

Flexibilidad
Sin estructura fija.

Soporte
Datos semiestructurados y no estructurados.

Ejemplos
Redis, Cassandra, MongoDB, Neo4j.

Tecnologías de Big Data: Real Time

📌 **Definición:**
Procesamiento continuo de datos a medida que se generan, con baja latencia.

🔍 **Casos de uso:**
Detección de fraude en tiempo real, monitoreo de redes sociales, sistemas de recomendación instantánea.

⚙️ **Tecnologías principales:**
Apache Kafka, Apache Flink, Apache Storm, Spark Streaming.

🧠 **Conceptos clave de Stream Processing**

•**Ventanas de tiempo (Windows):**
Permiten agrupar eventos en intervalos (por tiempo, conteo o sesiones) para procesarlos juntos.
Ej.: Sumar ventas cada 5 segundos.

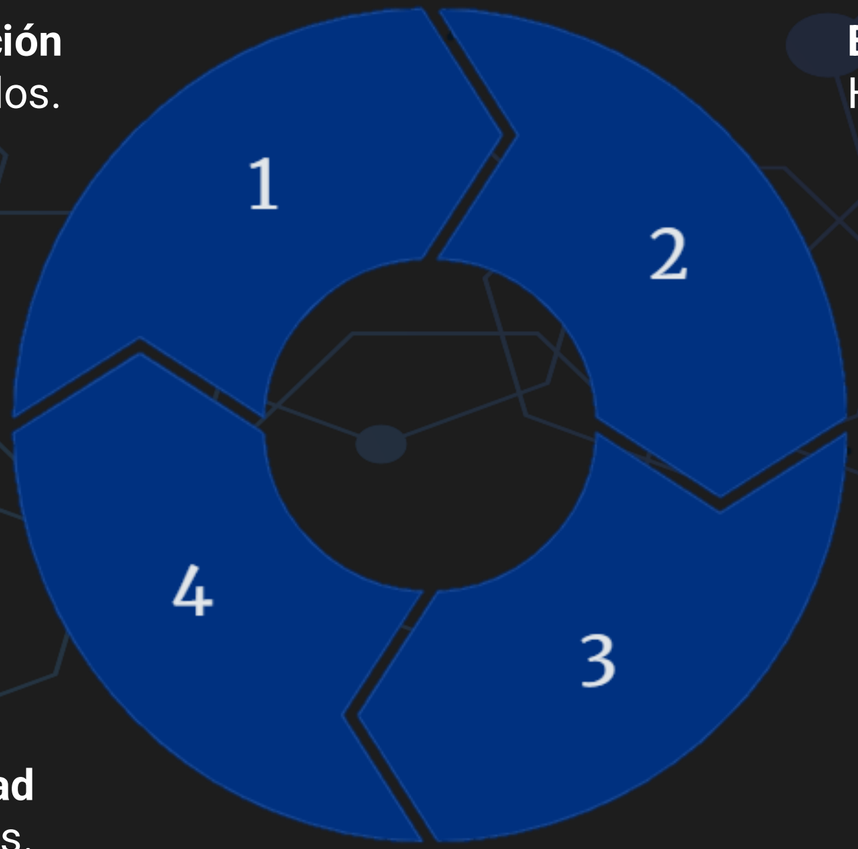
•**Checkpointing y tolerancia a fallos:**
Guardan el estado del sistema periódicamente para reiniciar en caso de fallo sin perder datos.

•**Exactly-once Semantics:**
Garantiza que cada evento se procese una única vez, incluso en fallos — esencial para integridad de datos financieros o críticos.

Tecnologías de Big Data: Almacenamiento Distribuido

Definición
Datos repartidos en múltiples nodos.

Ejemplos
HDFS, Amazon S3, Google Cloud Storage.



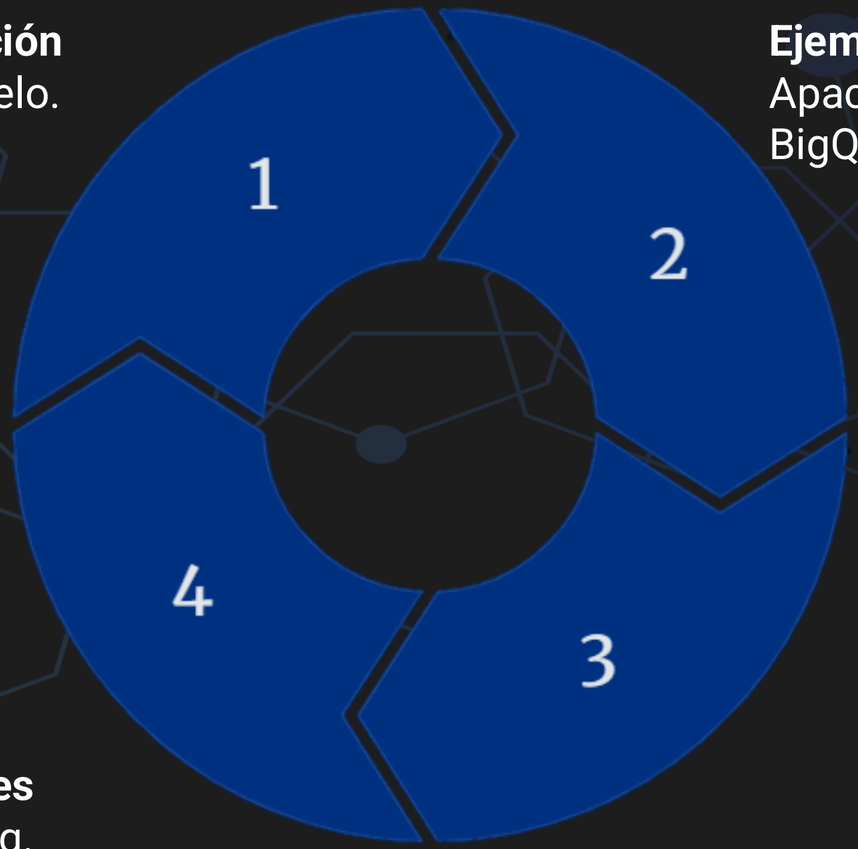
Escalabilidad
Agregar más nodos.

Características
Replicación y tolerancia a fallos.

Tecnologías de Big Data: Procesamiento Distribuido

Definición
Múltiples servidores trabajando en paralelo.

Ejemplos
Apache Hadoop, Apache Spark, Google BigQuery.



Enfoques
Batch Processing y Stream Processing.

Características
Dividir la carga de trabajo.

Comparativa: Batch vs Stream Processing

Característica	**Batch Processing**	**Stream Processing**
Latencia	Alta (procesamiento diferido)	Baja (procesamiento en tiempo real o casi real)
Frecuencia	Periódico (ej. una vez al día)	Continuo (eventos procesados al instante)
Fuentes de datos	Archivos grandes, bases de datos históricas	Flujos de eventos, sensores, APIs en tiempo real
Ejemplo típico	ETL nocturna, generación de reportes	Detección de fraude, monitoreo de redes sociales
Tecnologías comunes	Apache Hadoop, Spark (modo batch)	Apache Kafka, Apache Flink, Spark Streaming

Tecnologías de Big Data: Frameworks

- Apache Hadoop
- Apache Spark
- Apache Flink
- Apache Storm
- Dask

Arquitecturas típicas de Big Data

Arquitectura en Capas de Big Data

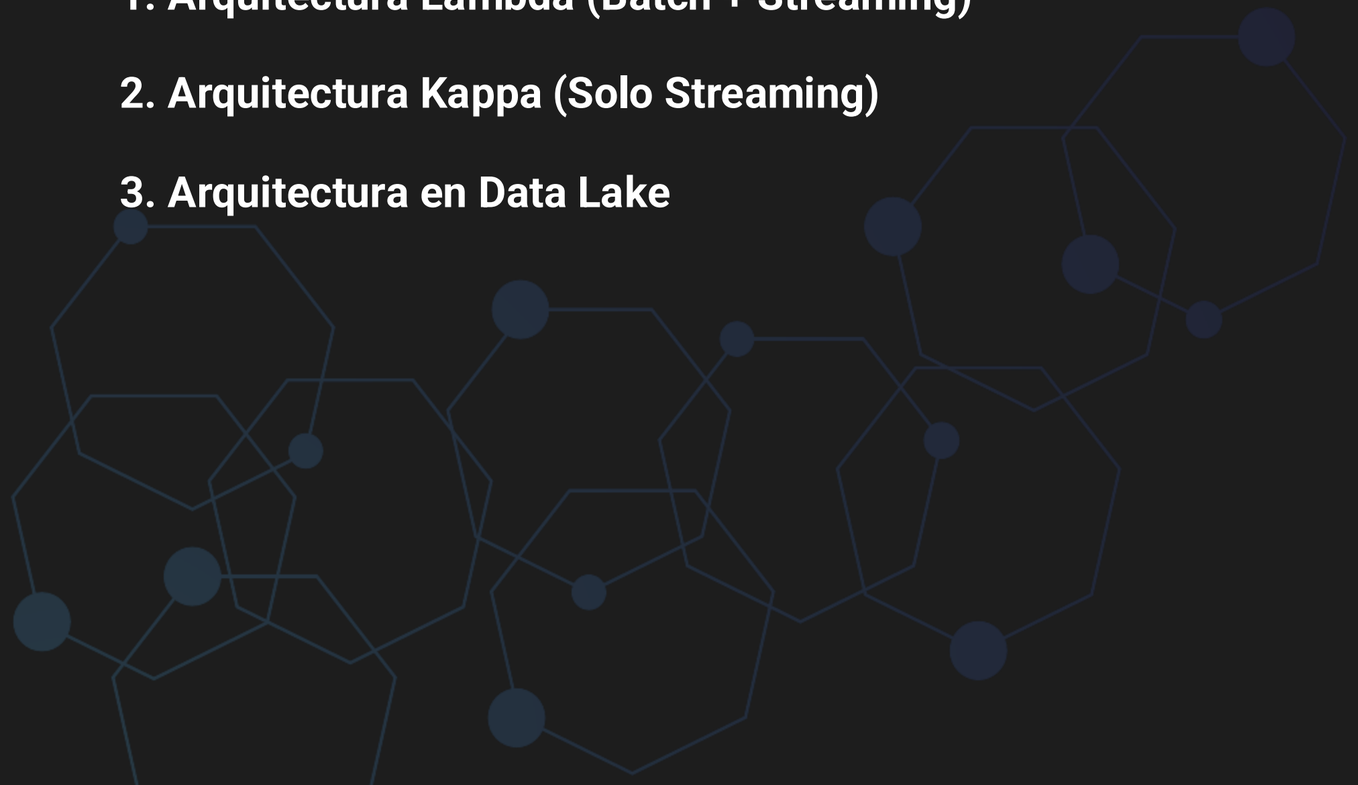
Una arquitectura típica de Big Data se divide en varias capas, cada una con funciones específicas:

1. **Capa de Ingesta de Datos**
2. **Capa de Almacenamiento**
3. **Capa de Procesamiento**
4. **Capa de Análisis y Machine Learning**
5. **Capa de Visualización e Interfaz de Usuario**

Modelos de Arquitectura en Big Data

Dependiendo del uso y necesidades, existen varias arquitecturas comunes:

1. **Arquitectura Lambda (Batch + Streaming)**
2. **Arquitectura Kappa (Solo Streaming)**
3. **Arquitectura en Data Lake**



Evolución de las Tecnologías

1. **Hadoop**
   Inicio del procesamiento distribuido.
2. **Spark**
   Procesamiento en memoria, más rápido.
3. **Otras**
   Framework, almacenamiento en la nube, NoSQL.

Ejemplos prácticos: HDFS y Spark SQL

📂 HDFS – Listar archivos en el sistema distribuido:

```
hdfs dfs -ls /user/datos/
```

Este comando muestra los archivos almacenados en la ruta /user/datos/ del sistema de archivos distribuido HDFS.

Ejemplos prácticos: HDFS y Spark SQL

🔎 Spark SQL – Consulta simple en PySpark:

```
# Crear una vista temporal desde un DataFrame
df = spark.read.csv("ventas.csv", header=True, inferSchema=True)
df.createOrReplaceTempView("ventas")

# Consulta SQL
spark.sql("SELECT producto, SUM(total) AS total_ventas FROM ventas GROUP BY producto").show()
```

Este ejemplo carga un archivo CSV y ejecuta una consulta SQL para obtener el total de ventas por producto.

Preguntas

Sección de preguntas

Fundamentos de

Big Data

Continúe con las actividades