Big Data y Hadoop

Data Science (CC3084) - 2026

Big Data y Hadoop

Semestre 02, 2026

Definición

Big Data engloba conjuntos extremadamente grandes y diversos de datos estructurados, no estructurados y semiestructurados que crecen exponencialmente a lo largo del tiempo.

La definición no depende de un número. Un conjunto de datos entra en el terreno de Big Data cuando las herramientas tradicionales dejan de servir: una sola máquina ya no lo almacena, una base de datos relacional ya no lo consulta en tiempo razonable, y una hoja de cálculo ni siquiera lo abre.

El cambio de escala

- Los métodos estadísticos clásicos suponen que los datos caben en memoria y llegan de forma ordenada.
- Los datos modernos llegan de sensores, aplicaciones, transacciones y redes sociales, sin pausa y sin esquema fijo.
- La respuesta no es una computadora más grande, sino muchas computadoras trabajando de forma coordinada.

La idea central

```
Muchos datos  ->  Muchas máquinas  ->  Procesamiento distribuido  ->  Decisiones
```

Todo el ecosistema de Big Data existe para resolver una sola tensión: los datos crecen más rápido que la capacidad de una sola máquina.

Las V del Big Data

Las características de Big Data se resumen tradicionalmente en un conjunto de dimensiones que empiezan con la letra V. Las tres originales eran volumen, velocidad y variedad; con el tiempo la lista creció.

Las V fundamentales

- Volumen: la cantidad de datos generados y almacenados, medida hoy en terabytes, petabytes y más.
- Velocidad: el ritmo al que los datos se generan y deben procesarse, a veces en tiempo real.
- Variedad: la diversidad de formatos, desde tablas hasta imágenes, audio, texto libre y registros de servidor.
- Veracidad: la confiabilidad de los datos, afectada por ruido, sesgo, duplicados y valores faltantes.

Las V de aprovechamiento

- Viabilidad: la capacidad real de una organización de usar los datos que recopila.
- Visualización: la necesidad de presentar los resultados de forma comprensible para quien decide.

- Valor: el beneficio concreto que se extrae de los datos, la razón última del esfuerzo.

Las V de calidad y riesgo

- Validez: la precisión y exactitud de los datos con respecto al uso previsto.
- Volatilidad: el tiempo durante el cual un dato sigue siendo relevante, y las transiciones rápidas e inesperadas.
- Variabilidad: cómo cambia el significado o la estructura de los datos a lo largo del tiempo.
- Vulnerabilidad: las medidas de seguridad necesarias para que los datos se procesen conforme a la legislación y a los deseos del cliente.

Volumen y validez no son lo mismo

Un conjunto enorme puede ser inútil. Tener más filas no corrige un instrumento mal calibrado ni un formulario mal diseñado. La escala amplifica tanto la señal como el error.

Fuentes de Big Data

Los datos masivos no vienen de un solo lugar. Reconocer la fuente ayuda a anticipar su formato, su velocidad y su calidad.

Fuentes principales

- Redes sociales: publicaciones, reacciones, comentarios, grafos de seguidores.
- Transacciones: compras, pagos, reservas, movimientos bancarios.
- Máquinas y sensores: internet de las cosas, telemetría, GPS, cámaras.
- Registros de sistemas: bitácoras de servidores, clics, sesiones de usuario.

- Documentos y multimedia: correos, contratos, imágenes, audio, video.
- Datos abiertos: censos, clima, indicadores públicos, datos gubernamentales.

Estructura de los datos

| Tipo | Descripción | Ejemplo | |---|---|---| | Estructurado | Esquema fijo, filas y columnas | Tabla de ventas | | Semiestructurado | Etiquetas o jerarquía, sin esquema rígido | JSON, XML, logs | | No estructurado | Sin esquema alguno | Texto libre, imágenes, audio |

La mayor parte del volumen que producen las organizaciones cae en las dos últimas categorías, y ahí es donde fallan las herramientas tradicionales.

Desafíos

Cada V se convierte en un problema de ingeniería concreto.

Volumen de datos

- Dónde almacenarlos cuando ya no caben en un disco.
- Cómo procesarlos cuando una sola CPU tardaría semanas.
- Cómo analizarlos cuando el algoritmo clásico supone que todo cabe en memoria.

Velocidad de generación

- Cómo procesar y analizar a la misma velocidad a la que llegan los datos.
- Cómo obtener información válida para tomar decisiones antes de que pierda vigencia.

- Qué hacer cuando el flujo supera temporalmente la capacidad de procesamiento.

Privacidad y seguridad

- Cómo proteger datos repartidos en cientos de nodos.
- Cómo asegurar la privacidad de las personas detrás de esos registros.
- Cómo cumplir con la legislación aplicable sin bloquear el análisis.

Calidad y talento

- Los datos llegan sucios, duplicados y con huecos, y limpiarlos consume la mayor parte del proyecto.
- Integrar fuentes heterogéneas exige acordar identificadores, unidades y significados.
- El costo de infraestructura y de personal capacitado sigue siendo una barrera real.

Tipos de Big Data

Una distinción práctica separa los datos por la función que cumplen dentro de la organización.

Big Data operacional

Son los datos cotidianos que el negocio genera al funcionar. Se caracterizan por alto volumen de escrituras, operaciones pequeñas y necesidad de disponibilidad constante.

Ejemplos

- Reservas en línea de boletos de tren, avión o cine.
- Compras en tiendas de comercio electrónico.
- Actividad en aplicaciones y redes sociales.

Big Data analítica

Son los datos que se procesan para apoyar decisiones. Se caracterizan por consultas complejas sobre grandes históricos y por agregaciones que cruzan muchas fuentes.

Ejemplos

- Comercialización de acciones en mercados financieros.
- Misiones espaciales donde cada bit de información es crucial.
- Pronóstico del tiempo.
- Monitoreo del estado de salud de un paciente.

Relación entre ambos

```
Operacional  ->  ingestión  ->  Analítica  ->  Decisión  ->  Operacional
```

Lo analítico se alimenta de lo operacional, y la decisión que produce vuelve a cambiar el sistema que genera los datos.

Principales Tecnologías

El ecosistema es amplio, pero se organiza en capas con responsabilidades distintas.

| Capa | Función | Herramientas | |----------------|--|----------------------------------| | Almacenamiento | Guardar datos distribuidos | HDFS,

S3, Cassandra, MongoDB | | Recursos | Repartir CPU y memoria del clúster | YARN, Mesos, Kubernetes | | Procesamiento | Ejecutar el cómputo distribuido | MapReduce, Spark, Flink | | Consulta | Analizar con lenguajes conocidos | Hive, Spark SQL, Presto | | Ingesta | Mover datos entre sistemas | Kafka, Sqoop, Flume | | Plataforma | Integrar todo el ciclo de vida | Databricks, BigQuery, Snowflake |

Esta lección se concentra en las dos piezas fundacionales del ecosistema, Hadoop y Spark, y cierra con Databricks como plataforma integrada.

Apache Hadoop

Hadoop es un framework de código abierto para almacenar y procesar grandes volúmenes de datos en clústeres de computadoras comunes. Su premisa es que el hardware falla, y el software debe asumirlo.

Principios de diseño

- Cada nodo del clúster tiene su propio disco, memoria, ancho de banda y procesamiento.
- Los datos entrantes se dividen en bloques individuales que se guardan en la capa de almacenamiento distribuido.
- El sistema asume que ninguna unidad de disco ni nodo esclavo es confiable.
- Cada conjunto de datos se replica en varios nodos del clúster.
- El nodo maestro conserva los metadatos de cada bloque y de todas sus réplicas.

Las tres capas

```
┌─────────────────────────────────────────────────────────┐
│  Aplicaciones     Hive · Spark · Sqoop · MapReduce  │
```

```
┌──────────────────────────────────────────────────────┐
│  Cómputo          YARN, reparte los recursos         │
├──────────────────────────────────────────────────────┤
│  Almacenamiento   HDFS, guarda los bloques           │
├──────────────────────────────────────────────────────┤
│  Hardware         nodos con RAM + CPU + disco        │
└──────────────────────────────────────────────────────┘
```

Separar almacenamiento, recursos y procesamiento permite que cada capa evolucione por su cuenta, y explica por qué Spark pudo reemplazar a MapReduce sin tocar HDFS.

Mover el cómputo, no los datos

La idea clave de Hadoop es que copiar un petabyte por la red es carísimo, pero enviar un programa de unos kilobytes a la máquina donde ya está el dato es barato. Las tareas se distribuyen lo más cerca posible de los servidores donde residen los bloques.

HDFS

HDFS es la capa de almacenamiento distribuido de Hadoop. Toma un archivo enorme, lo corta en bloques y los reparte por el clúster.

Cómo funciona

- Un archivo se divide en bloques de tamaño fijo, típicamente 128 MB.
- Cada bloque se replica, por defecto tres veces, en nodos distintos.
- Las réplicas se colocan en racks diferentes para sobrevivir a la caída de un rack completo.
- Los nodos de datos envían una señal periódica al nodo maestro para indicar que siguen vivos.

Componentes

| Componente | Rol | |------------|---| | NameNode | Nodo maestro, guarda los metadatos y sabe dónde vive cada bloque | | DataNode | Nodo esclavo, almacena los bloques reales y reporta su estado |

El NameNode nunca guarda datos de usuario. Guarda el mapa. Si se pierde el mapa, los bloques siguen ahí pero nadie sabe cómo reconstruir el archivo, y por eso el NameNode es el punto crítico del diseño.

Tolerancia a fallos

Cuando un DataNode deja de responder, el NameNode detecta que ciertos bloques bajaron de tres réplicas a dos y ordena copiarlos a otro nodo. El usuario nunca se entera. La falla es esperada, no excepcional.

YARN

YARN es el negociador de recursos del clúster, y su nombre viene de Yet Another Resource Negotiator. Su trabajo es que muchas aplicaciones y usuarios compartan los mismos nodos sin estorbarse.

Componentes

| Componente | Rol | |--------------------|---| | ResourceManager | Decide qué aplicación recibe qué recursos y cuándo | | NodeManager | Vigila los recursos de su nodo y reporta al ResourceManager | | ApplicationMaster | Coordina una aplicación concreta y pide recursos para ella | | Container | Paquete de memoria y CPU donde corre efectivamente una tarea |

Flujo de trabajo

1. Una aplicación cliente envía una solicitud al ResourceManager.
2. El ResourceManager le indica a un NodeManager que inicie un ApplicationMaster para esa solicitud, dentro de un contenedor.
3. El ApplicationMaster se registra con el ResourceManager, consulta al NameNode dónde están los bloques necesarios y calcula cuántas tareas hacen falta.
4. El ApplicationMaster solicita los recursos al ResourceManager y sigue comunicando sus necesidades durante todo el ciclo de vida.
5. El ResourceManager encola esa solicitud junto con las de los demás ApplicationMasters y va liberando recursos conforme se desocupan.
6. El ApplicationMaster contacta al NodeManager del nodo asignado y le pide crear un contenedor con las variables, los tokens de autenticación y el comando a ejecutar.
7. El ApplicationMaster supervisa el proceso y lo reinicia si falla. Después de cuatro intentos fallidos, el trabajo completo falla.

MapReduce

MapReduce es el modelo de programación original de Hadoop. Procesa datos dispersos en el clúster mediante tres fases: los datos se mapean, se barajan y se reducen a un resultado agregado.

Fase de mapa

- El proceso de mapeo toma expresiones lógicas individuales de los datos almacenados en los bloques de HDFS.
- Esas expresiones pueden abarcar varios bloques y se llaman divisiones de entrada.

- Las divisiones se entregan al proceso de mapeo como pares clave-valor.
- Cada tarea de mapeo analiza sus pares y produce un conjunto nuevo de pares clave-valor, distinto de la entrada original.

Fase de mezcla y clasificación

- Los resultados de todas las tareas de mapa se copian a los nodos reductores, y esa copia es el único intercambio de datos entre nodos durante todo el trabajo.
- Los pares se agrupan por clave y se ordenan, porque el reductor necesita ver juntos todos los valores de una misma clave.
- La mezcla y el ordenamiento se ejecutan en paralelo, mientras las salidas del mapa se van recuperando.

Fase de reducción

- Las salidas del mapa llegan mezcladas y ordenadas en un único archivo de entrada dentro del nodo reductor.
- La función de reducción agrega los valores según su clave.
- El resultado es un nuevo par clave-valor que se almacena, por defecto, en HDFS.
- Todas las tareas de reducción se ejecutan de forma simultánea e independiente, y la fase de reducción es opcional.

Ejemplo, contar palabras

```
Entrada     "el gato y el perro"

Mapa        (el,1) (gato,1) (y,1) (el,1) (perro,1)

Mezcla      (el,[1,1])  (gato,[1])  (y,[1])  (perro,[1])
```

```
Reduce      (el,2)  (gato,1)  (y,1)  (perro,1)
```

El mismo patrón sirve para contar palabras en un tuit o en el archivo completo de Wikipedia. Lo único que cambia es la cantidad de nodos.

El límite de MapReduce

Cada trabajo escribe su resultado intermedio en disco antes de que empiece el siguiente. Para un proceso de un paso esto es aceptable. Para un algoritmo iterativo que recorre los mismos datos cincuenta veces, el costo de escribir y leer disco domina el tiempo total. Ese límite es exactamente lo que Spark vino a resolver.

Resumen

- Big Data describe conjuntos de datos que crecen más rápido de lo que una sola máquina puede almacenar o procesar.
- Las V resumen sus características, desde las tres originales de volumen, velocidad y variedad hasta las de calidad, riesgo y aprovechamiento.
- Las fuentes van de redes sociales y sensores a transacciones y registros de sistemas, y la mayoría produce datos no estructurados.
- Los desafíos son almacenamiento, velocidad de procesamiento, privacidad, seguridad y calidad de los datos.
- Big Data operacional es lo que el negocio genera al funcionar, y Big Data analítica es lo que se procesa para decidir.
- Hadoop resuelve el problema separando almacenamiento en HDFS, recursos en YARN y procesamiento en MapReduce.

- HDFS parte los archivos en bloques replicados y asume que el hardware va a fallar.
- YARN reparte contenedores entre aplicaciones mediante el ResourceManager, los NodeManagers y un ApplicationMaster por trabajo.
- MapReduce procesa en tres fases, mapa, mezcla y clasificación, y reducción, pero paga el costo de escribir a disco entre etapas.
- Ese último límite es el punto de partida de la siguiente lección.

Bibliografía

- Google Cloud. Qué es Big Data. https://cloud.google.com/learn/what-is-big-data?hl=es
- Características (V) del Big Data en la Industria. https://www.researchgate.net/publication/353176808
- Desafíos del Big Data. https://www.sycod.com/blog/desafios-del-big-data/
- Arquitectura de Hadoop. https://techvidvan.com/tutorials/hadoop-architecture/
- Apache Hadoop Architecture Explained. https://phoenixnap.com/kb/apache-hadoop-architecture-explained