
Maestría en Analítica de Datos

BigData-Sesión 11

# Procesamiento de Big Data con MapReduce en Hadoop

Luis Fernando Castellanos Guarin

2025

UNIVERSIDAD CENTRAL

YO ♥ UC

**Un clúster de servidores o una visualización abstracta de grandes datos fluyendo)**

UNIVERSIDAD CENTRAL

# Introducción al Procesamiento de Big Data con MapReduce en Hadoop

Desbloqueando el poder de los datos a escala

# Agenda

UNIVERSIDAD CENTRAL

- La era del Big Data y sus desafíos.
- Nacimiento de MapReduce: Un poco de historia.
- Conceptos Fundamentales de MapReduce: Map y Reduce.
- La magia de Shuffle & Sort.
- Arquitectura de MapReduce en Acción.
- Hadoop: El ecosistema para MapReduce.
- HDFS: Almacenando Big Data para MapReduce.
- Funcionamiento de MapReduce en un clúster Hadoop.
- Beneficios de usar MapReduce.
- Limitaciones de MapReduce (y la evolución).
- MapReduce con Python: Herramientas y enfoques.
- Casos de Uso de MapReduce.
- Más allá de MapReduce (Breve mención a otros frameworks).
- Preparándonos para el Ejercicio Práctico.
- Preguntas y Discusión.

# La Era del Big Data y sus Desafíos

UNIVERSIDAD CENTRAL

**El Diluvio de Datos:**

- Vivimos en un mundo donde se generan exabytes de datos constantemente (redes sociales, IoT, transacciones, sensores, etc.).
- **Volumen**: Cantidades masivas de datos.
- **Velocidad**: Datos que llegan a gran velocidad.
- **Variedad**: Datos estructurados, semi-estructurados y no estructurados.
- **Veracidad**: La calidad y confianza de los datos.
- **Valor**: La capacidad de extraer conocimiento útil.

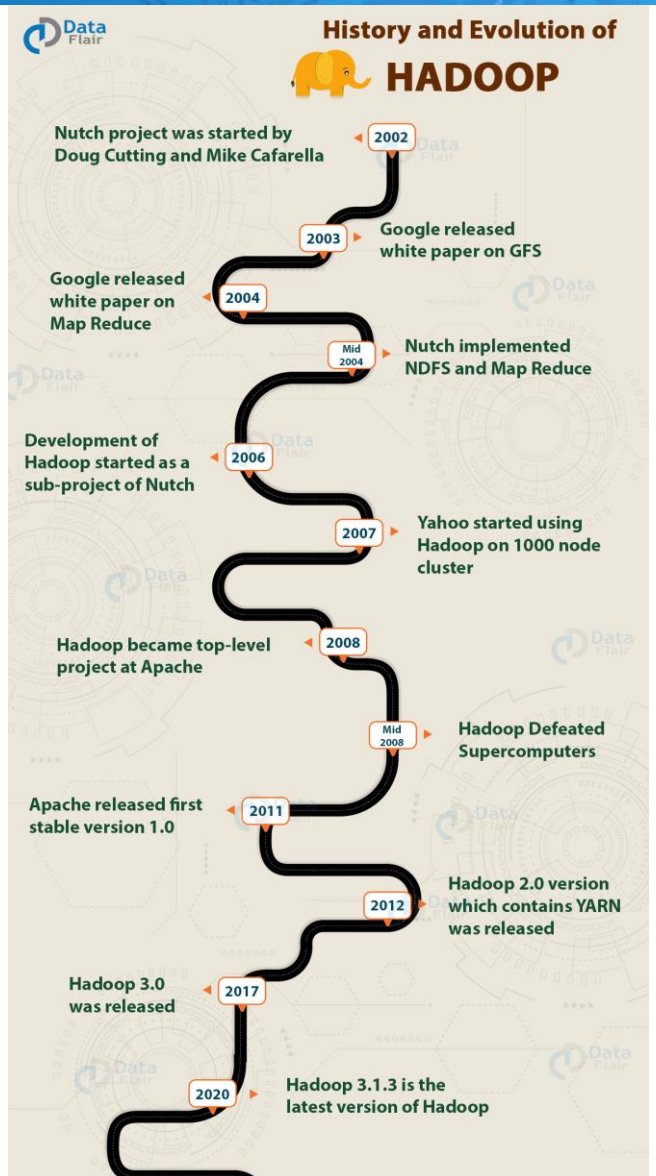
**El Desafío:** Las herramientas y bases de datos tradicionales no son suficientes para almacenar, procesar y analizar esta escala de datos de manera eficiente.

# Nacimiento de MapReduce: Un Poco de Historia

UNIVERSIDAD CENTRAL

**La Chispa en Google:**

1. A principios de la década de 2000, Google enfrentaba desafíos masivos para indexar la web y procesar enormes cantidades de datos.
2. Jeff Dean y Sanjay Ghemawat publicaron en 2004 el paper "MapReduce: Simplified Data Processing on Large Clusters".
3. Inspirado en funciones de programación funcional (**map y reduce**).
4. Propuso un modelo de programación simple pero poderoso para procesar datos en paralelo en clústeres distribuidos.
5. Sentó las bases para el procesamiento distribuido de Big Data.



# ¿pero Qué es MapReduce?

UNIVERSIDAD CENTRAL

- Modelo de programación para procesar grandes cantidades de datos en paralelo.
- Divide el trabajo en dos fases principales: **Map** y **Reduce**.
- Inspirado en funciones clásicas de programación funcional:
  - **Map:** Aplica una función a cada elemento de una colección y devuelve una nueva colección.
  - **Reduce (fold):** Agrega todos los elementos de una colección usando una operación acumulativa (como suma, concatenación, etc.).
- En el contexto de Big Data, MapReduce adapta estas ideas para distribuir el trabajo entre múltiples nodos y procesar datos a gran escala.

# Ejemplos de map y reduce en Python

UNIVERSIDAD CENTRAL

- Para entender MapReduce, exploremos primero los conceptos en Python puro:

```
# Ejemplo 1: Map - Elevar al cuadrado cada númeronumeros = [1, 2, 3, 4, 5]
cuadrados = list(map(lambda x: x ** 2, numeros))
print(cuadrados)  # [1, 4, 9, 16, 25]
```

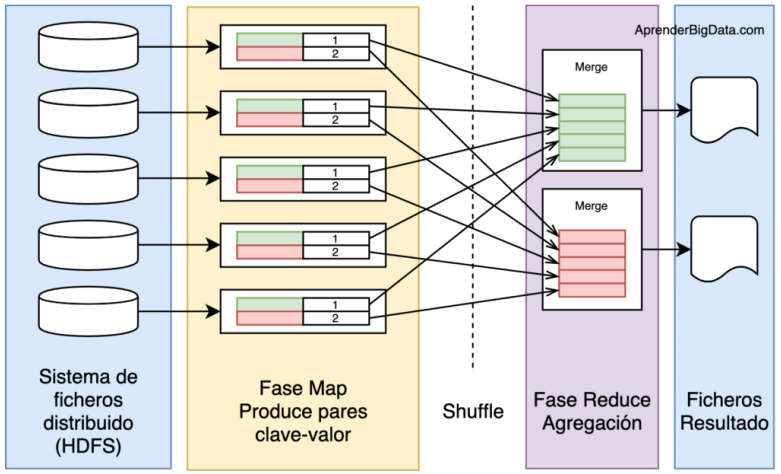
```
# Ejemplo 2: Reduce - Sumar todos los números de una lista
from functools import reduce
suma_total = reduce(lambda x, y: x + y, numeros)
print(suma_total)  # 15
```

# Conceptos Fundamentales de MapReduce: Reduce

UNIVERSIDAD CENTRAL

**La Fase Map: Transformando Datos:**

- La fase Map es la primera etapa del procesamiento MapReduce.
- Toma datos de entrada (generalmente pares ***<key, value>***).
- Aplica una función de mapeo definida por el usuario a cada par de entrada de forma independiente y paralela.
- Emite un nuevo conjunto de pares ***<intermediate key, intermediate value>***.
- Objetivo: Transformar y filtrar los datos brutos en un formato intermedio relevante para el problema a resolver.Ejemplo: En conteo de palabras, el mapper recibe una línea de texto y emite pares ***<word, 1>*** por cada palabra.

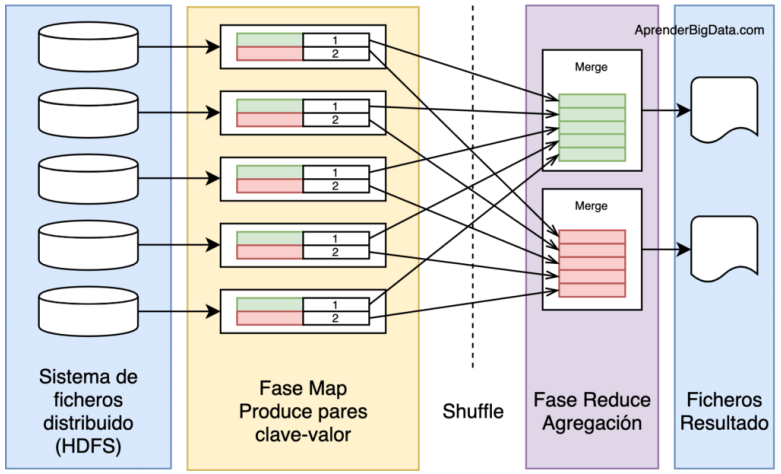


# La Magia de Shuffle & Sort

UNIVERSIDAD CENTRAL

## Conectando Map y Reduce

- Esta es la fase intermedia crucial entre Map y Reduce.
- **Shuffle**: El framework MapReduce (en Hadoop, gestionado por YARN y el propio framework MapReduce) se encarga de recopilar la salida de todas las tareas Map.
- **Sort**: Los pares intermedios se ordenan por clave.
- **Grouping**: Se agrupan todos los valores que tienen la misma clave intermedia.
- Estos datos agrupados se envían a la tarea Reduce correspondiente.
- **Importancia**: Permite que la función Reduce trabaje con todos los valores de una clave específica, independientemente de qué tarea Map los generó. Es la clave para la agregación distribuida.



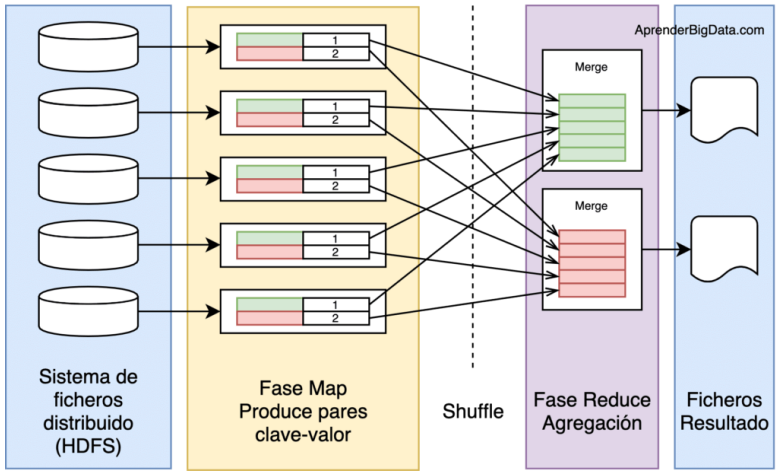
# Arquitectura de MapReduce en Acción

UNIVERSIDAD CENTRAL

## Cómo Fluye el Procesamiento

Esta es la fase intermedia crucial entre Map y Reduce.

- **Input Splitting:** Los datos de entrada se dividen en fragmentos lógicos (Input Splits). Cada split es procesado por una tarea Map.
- **Map Phase:** Múltiples tareas Map se ejecutan en paralelo en diferentes nodos del clúster, procesando sus respectivos Input Splits y generando pares intermedios.
- **Shuffle & Sort Phase:** El framework recoge los pares intermedios, los agrupa por clave y los ordena.
- **Reduce Phase:** Múltiples tareas Reduce se ejecutan en paralelo, recibiendo los datos agrupados para sus claves asignadas y generando la salida final.
- **Output:** Los resultados de las tareas Reduce se escriben en el sistema de archivos de salida.



# Hadoop: El Ecosistema para MapReduce

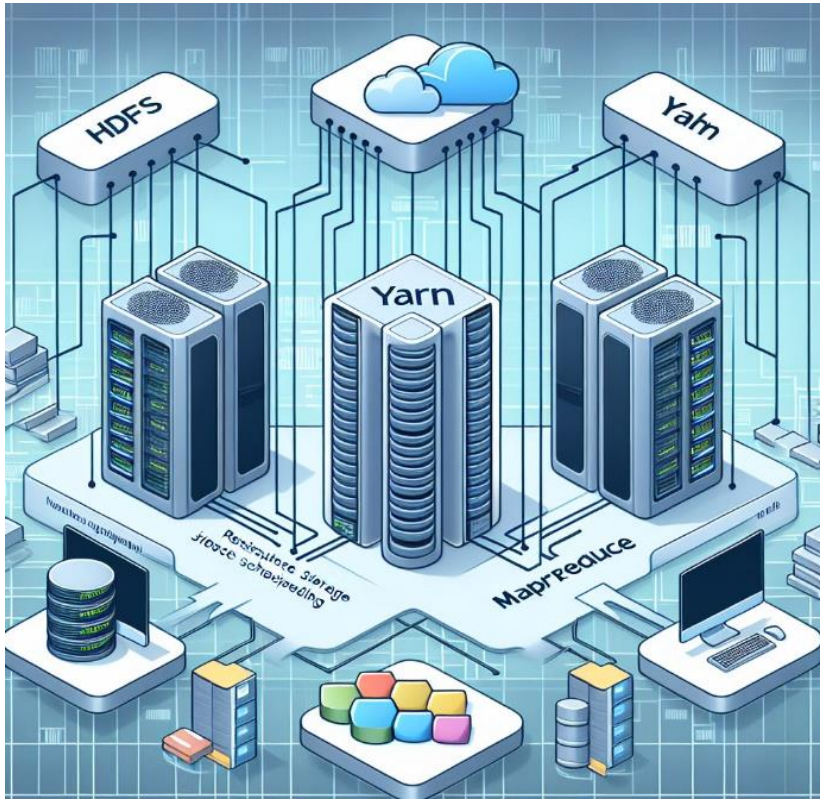
UNIVERSIDAD CENTRAL

## La Plataforma que Potencia MapReduce

**Apache Hadoop:** Un framework de código abierto para almacenamiento distribuido y procesamiento distribuido de grandes conjuntos de datos en clústeres de computadoras.

**Componentes Clave de Hadoop para MapReduce:**

- **HDFS (Hadoop Distributed File System):** Sistema de archivos distribuido tolerante a fallos para almacenar los datos de entrada y salida.
- **YARN (Yet Another Resource Negotiator):** Gestiona los recursos del clúster (CPU, memoria) y planifica la ejecución de las aplicaciones (incluidos los trabajos MapReduce).
- **MapReduce:** El motor de procesamiento en sí.



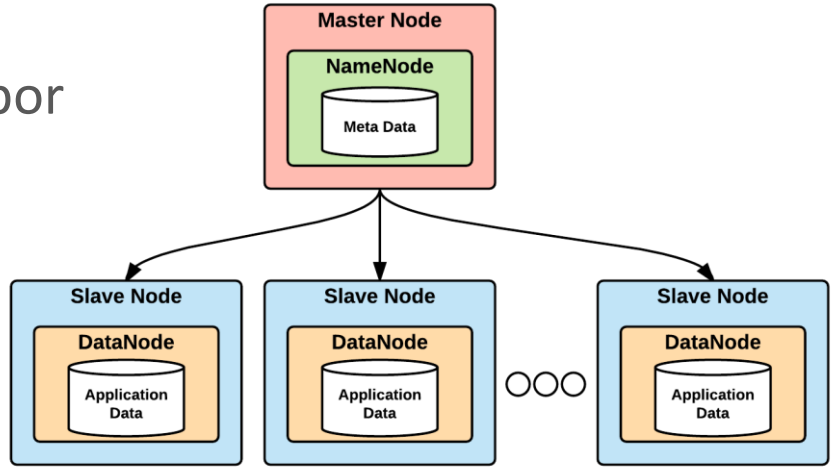

**Hadoop** proporciona la infraestructura necesaria para escalar MapReduce a través de cientos o miles de nodos.

# HDFS: Almacenando Big Data para MapReduce

UNIVERSIDAD CENTRAL

## El Sistema de Archivos Distribuido de Hadoop

- Diseñado para almacenar archivos muy grandes a través de múltiples máquinas.
- **Tolerancia a Fallos:** Replica los bloques de datos en varios nodos (por defecto, 3 copias). Si un nodo falla, los datos siguen disponibles en otras réplicas.
- **Optimizado para Lectura Secuencial:** Eficiente para leer grandes archivos de principio a fin, que es la forma en que MapReduce procesa los datos.
- **Locality:** Permite que las tareas de procesamiento (Map) se ejecuten en los nodos donde residen los datos, minimizando el movimiento de datos a través de la red. Esto es crucial para el rendimiento en Big Data.



# Funcionamiento de MapReduce en un Clúster Hadoop

UNIVERSIDAD CENTRAL

## Orquestando el Procesamiento Distribuido

1. **El Cliente** envía el trabajo MapReduce (código Map, código Reduce, configuración) al ResourceManager de YARN.
2. **ResourceManager** asigna un ApplicationMaster para el trabajo.
3. **ApplicationMaster** interactúa con el ResourceManager para solicitar recursos (contenedores) en los NodeManagers del clúster.
4. **ApplicationMaster** divide el trabajo en tareas Map y Reduce basadas en los Input Splits de HDFS.
5. **ApplicationMaster** envía las tareas Map a los NodeManagers que tienen los datos localmente (Data Locality).
6. **Los NodeManagers** lanzan los procesos (contenedores) para ejecutar las tareas Map.
7. Las tareas Map leen datos de HDFS, procesan y escriben los pares intermedios en el disco local de los nodos worker.
8. La fase **Shuffle & Sort** ocurre, moviendo y agrupando los datos intermedios hacia los nodos donde se ejecutarán las tareas Reduce.
9. **ApplicationMaster** envía las tareas Reduce a los NodeManagers.
10. Las tareas Reduce leen los datos intermedios agrupados, procesan y escriben la salida final en HDFS.

# Beneficios de Usar MapReduce

UNIVERSIDAD CENTRAL

## ¿Por qué MapReduce?

1. **Escalabilidad:** Permite procesar conjuntos de datos masivos distribuyendo la carga a través de un clúster de máquinas.
2. **Tolerancia a Fallos:** Si una tarea o nodo falla, el framework puede reiniciar la tarea en otro lugar sin perder el progreso general del trabajo.
3. **Procesamiento Paralelo:** Las tareas Map y Reduce se ejecutan simultáneamente en muchos nodos, lo que reduce significativamente el tiempo de procesamiento.
4. **Abstracción:** Simplifica la programación distribuida al proporcionar un modelo simple (Map y Reduce) y encargarse de las complejidades subyacentes (distribución de datos, tolerancia a fallos, programación de tareas).
5. **Costo-Efectividad:** Se ejecuta en hardware commodity (servidores estándar y más económicos) en lugar de sistemas de gama alta costosos.

# Limitaciones de MapReduce (y la Evolución)

UNIVERSIDAD CENTRAL

## ¿Cuándo MapReduce no es la Mejor Opción?

- **Procesamiento por Lotes:** Diseñado principalmente para trabajos por lotes, no para consultas interactivas o procesamiento en tiempo real de baja latencia.
- **Iteraciones:** Ineficiente para algoritmos iterativos (comunes en Machine Learning) ya que cada iteración requiere leer y escribir datos en HDFS.
- **Overhead de Disco:** La escritura y lectura constante de datos intermedios en disco durante Shuffle puede ser un cuello de botella de rendimiento.
- **Complejidad de Programación:** Aunque simplifica la distribución, escribir trabajos MapReduce complejos aún puede ser laborioso compared con frameworks más modernos.

**La Evolución:** Frameworks como Apache Spark abordan muchas de estas limitaciones al realizar el procesamiento en memoria y ofrecer modelos de programación más flexibles (aunque MapReduce sigue siendo fundamental para entender los conceptos).

# MapReduce con Python: Herramientas y Enfoques

UNIVERSIDAD CENTRAL

## Haciendo MapReduce accesible con Python

Aunque Hadoop MapReduce está escrito en Java, se pueden escribir trabajos MapReduce en otros lenguajes, incluido Python, utilizando **Hadoop Streaming**.

- **Hadoop Streaming:** Una utilidad que permite usar cualquier ejecutable o script (como scripts de Python) como mappers y reducers. La comunicación se realiza a través de la entrada y salida estándar ***(stdin/stdout).***
- **Librerías de Python:**
  - **mrjob:** Una popular librería de Yelp que facilita escribir y ejecutar trabajos MapReduce en Python, tanto localmente como en clústeres Hadoop o en la nube **(AWS EMR)**.
  - **Simulaciones Locales:** Para aprender los conceptos, podemos simular el comportamiento de Map y Reduce en Python sin necesidad de un clúster Hadoop completo.

# Preparándonos para el Ejercicio Práctico

UNIVERSIDAD CENTRAL

## Manos a la Obra en Google Colab

- Utilizaremos Google Colab para simular un entorno MapReduce simplificado en Python.
- Nos enfocaremos en implementar las funciones Map y Reduce para un problema clásico:

**Conteo de Palabras (Word Count).**

- Entenderemos cómo los datos fluyen entre las etapas Map y Reduce.

¡Prepárense para codificar y ver MapReduce en acción a pequeña escala!

# Ejercicio Práctico en Google Colab: Conteo de Palabras con MapReduce en Python

El ejercicio haremos el paso a paso para implementar un simple trabajo de Conteo de Palabras utilizando la lógica de **MapReduce** en un entorno simulado en Google Colab.

No necesitaremos un clúster Hadoop real, sino que implementaremos las fases de Map, Shuffle y Reduce en Python puro para entender el flujo.

**Objetivo:** Contar la frecuencia de cada palabra en un conjunto de documentos de texto.
**Dataset:** Crearemos un conjunto de datos de ejemplo simple directamente en Colab.

UNIVERSIDAD CENTRAL

YO ♥ UC

# Gracias