# Introdución a MapReduce

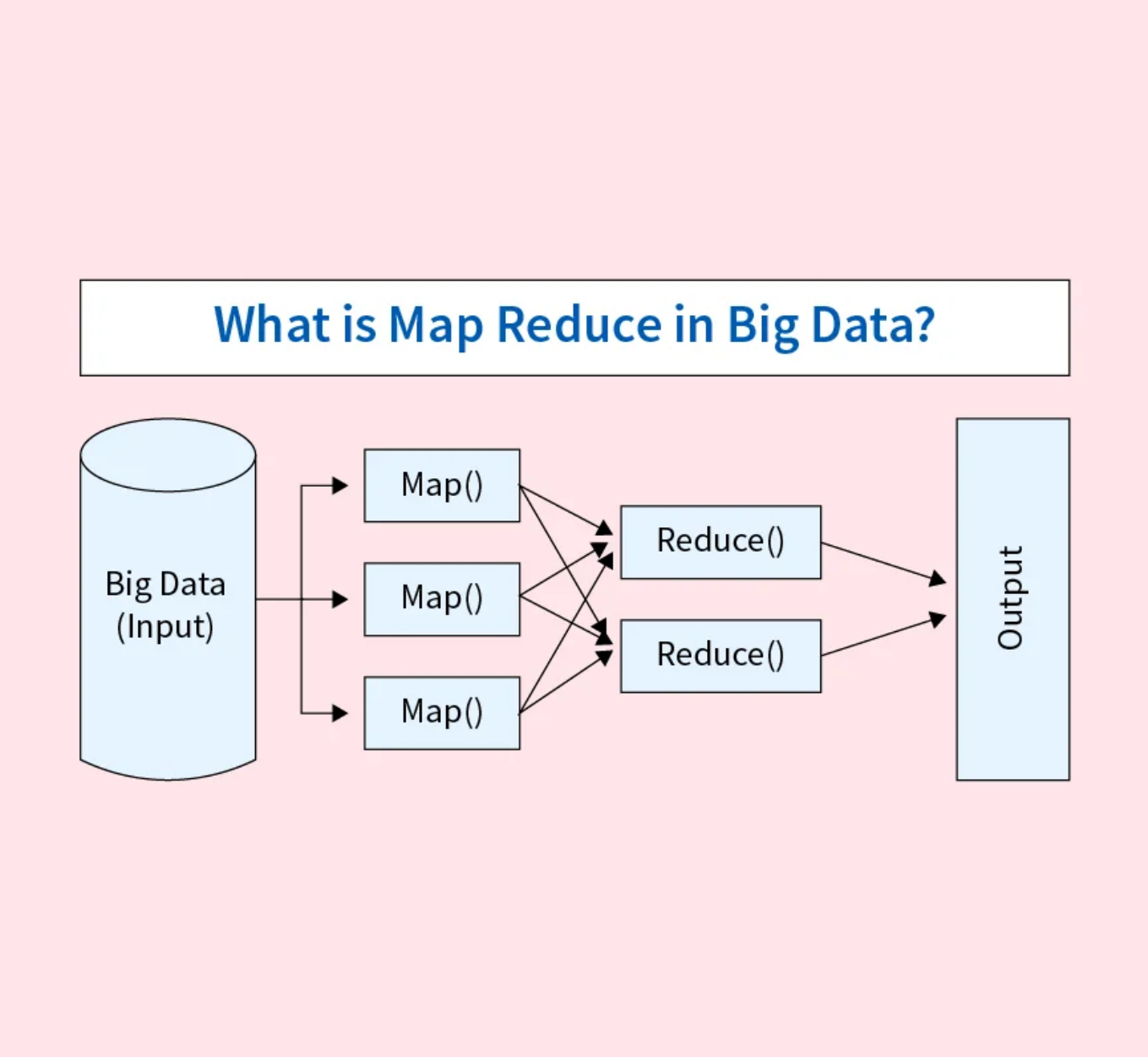
## Que é MapReduce

**MapReduce** é un modelo de programación deseñado para procesar grandes volumes de datos de maneira distribuída nun clúster. Basease na idea de dividir un problema grande en tarefas pequenas que poden executarse en paralelo.

O modelo estrutúrase en dúas fases principais:

- **Map**: transforma os datos de entrada en pares *(clave, valor)*.
- **Reduce**: agrupa os valores asociados a unha mesma clave e combínaos para obter un resultado final.

Este modelo é especialmente axeitado para operacións como contaxes, agregacións, estatísticas ou construción de índices.

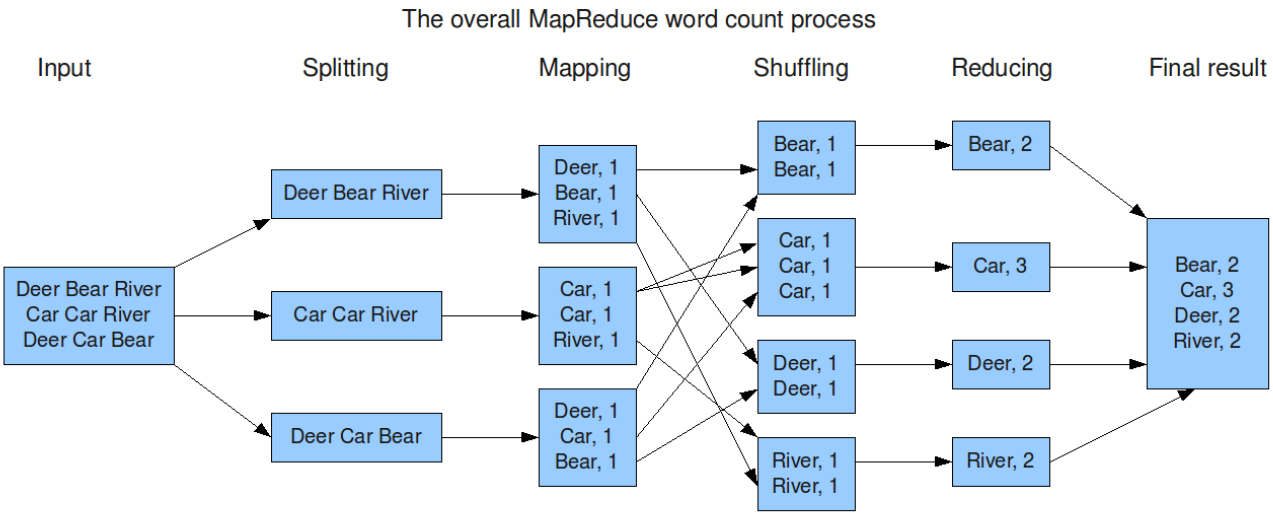



## Como funciona MapReduce internamente

1. **Input split**
    - Os datos de entrada divídense en fragmentos (*splits*).
    - Cada split é procesado por unha tarefa *map*.
2. **Fase Map**
    - O mapper le rexistros (normalmente liñas).
    - Por cada rexistro pode emitir cero, un ou varios pares *(clave, valor)*.
    - Exemplo: `(palabra, 1)` en WordCount.
3. **Combiner (opcional)**
    - Execútase localmente no nodo do mapper.
    - Reduce o volume de datos enviados pola rede.
    - Útil para operacións asociativas e conmutativas (como sumas).
4. **Shuffle e Sort**
    - Hadoop redistribúe os pares *(k, v)* para que todas as claves iguais cheguen ao mesmo reducer.
    - Os datos chegan ordenados e agrupados por clave.
5. **Fase Reduce**
    - O reducer recibe `(clave, [valores])`.
    - Aplica a función de redución e xera o resultado final.
6. **Output**
    - Os resultados escríbense en HDFS, en ficheiros `part-r-xxxxx`.

---

# Exemplo 1: WordCount co jar de exemplos de Hadoop



### Preparar os datos de entrada

```
hdfs dfs -mkdir -p /user/$USER/wc/input
hdfs dfs -put texto.txt /user/$USER/wc/input/
```

### Executar WordCount

```
export EXJAR=$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar

hdfs dfs -rm -r -f /user/$USER/wc/output

hadoop jar $EXJAR wordcount   /user/$USER/wc/input   /user/$USER/wc/output
```

### Consultar resultados

```
hdfs dfs -cat /user/$USER/wc/output/part-r-00000 | head
```

---

## Exemplo 2: WordCount con Python e Hadoop Streaming

### Mapper (mapper.py)

```
#!/usr/bin/env python3
import sys
import re

word_re = re.compile(r"[A-Za-zÀ-ÖØ-öø-ÿ0-9']+")

for line in sys.stdin:
    for w in word_re.findall(line.lower()):
        print(f"{w}\t1")
```

### Reducer (reducer.py)

```
#!/usr/bin/env python3
import sys

current_word = None
current_count = 0
```

```
for line in sys.stdin:
    word, count = line.strip().split("\t")
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count

if current_word:
    print(f"{current_word}\t{current_count}")
```

## Dar permisos

```
chmod +x mapper.py reducer.py
```

## Proba en local

```
cat texto.txt | ./mapper.py | sort | ./reducer.py
```

## Executar en Hadoop Streaming

```
export STREAMJAR=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar

hdfs dfs -rm -r -f /user/$USER/wc_stream/output

hadoop jar $STREAMJAR   -files mapper.py,reducer.py   -mapper mapper.py   -reducer
reducer.py   -input /user/$USER/wc_stream/input   -output
/user/$USER/wc_stream/output
```

## Ver resultados

```
hdfs dfs -cat /user/$USER/wc_stream/output/part-* | head
```

---

## Notas finais

- O número de reducers pódese configurar con:

```
-D mapreduce.job.reduces=2
```

- WordCount é un exemplo clásico porque mostra claramente o patrón Map → Shuffle → Reduce.
- Hadoop Streaming permite empregar calquera linguaxe que lea de stdin e escriba en stdout.