

📄 Apuntes completos de Map Reduce con Python + Hadoop Streaming Pensados para que alguien que parte de 0 pueda entender la filosofía de Map Reduce y practicar con una batería grande de ejercicios tipo-examen. Cada ejercicio incluye los dos programas (mapper.py y reducer.py) listos para lanzarse con:

```
cat fichero.data | ./mapper.py | sort | ./reducer.py
```

## 0. Tabla de contenido

1. ¿Qué es Map Reduce y por qué se creó?
2. Anatomía de un job
3. Hadoop Streaming en la práctica (arquitectura mínima)
4. Formato de fichero y convenciones
5. Patrones de diseño para mappers y reducers
6. Recetas de depuración rápida
7. Colección grande de ejercicios con solución
8. Trucos de rendimiento y preguntas de teoría
9. Checklist final antes del examen

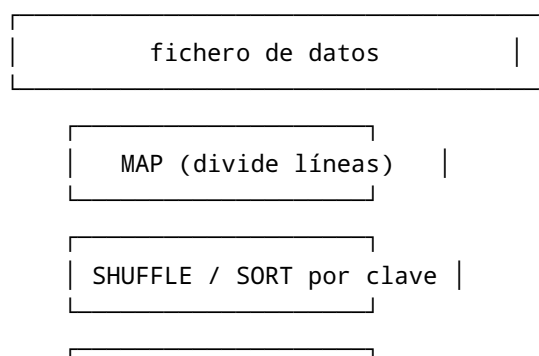
## 1. ¿Qué es Map Reduce?

Map Reduce es un patrón de programación creado originalmente en Google (Dean & Ghemawat, 2004) para procesar grandes volúmenes de datos distribuidos. Se basa en dos ideas:

Fase	Input	Output	Función
Map	Split de datos (ej. líneas)	\<clave, valor>	Transformar y emitir
Reduce	Todos los pares con misma clave	Resultado agregado	Combinar

La fase intermedia (Shuffle + Sort) es la “magia” que reagrupa y reordena los datos por clave.

## 2. Anatomía de un job MapReduce



```
| REDUCE (agrupa claves) |
```

### 3. Hadoop Streaming

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming*.jar \  
-input /ruta/ventas.tsv \  
-output /salida/e1 \  
-mapper m_e1.py -reducer r_e1.py \  
-file m_e1.py -file r_e1.py
```

En local se puede simular con:

```
cat localstore.data | ./m.py | sort | ./r.py
```

### 4. Formato del fichero de entrada

Col	Campo	Ejemplo
0	ID	315
1	Proveedor	Cafes el amanecer
2	Tipo de envío (1-3)	2
3	Peso (kg)	87
4	Coste de envío (€)	56
5	Fecha (dd/mm/aaaa)	16/05/2004
6	Origen	Madrid
7	Destino	Sevilla
8	Importe de venta	450.32

### 5. Patrones de diseño

Necesitas...	Mapper emite	Reducer hace
Suma por clave	print(k,v)	acumula y emite total
Media por clave	print(k,v)	suma y divide
Top-N global	print(k,1)	cuenta y filtra

Necesitas...	Mapper emite	Reducer hace
Join por campo	prefijo de fuente	agrupa y combina
Filtro complejo	no emite	---
Histograma / buckets	print(rango,1)	suma por bucket

## 6. Depuración rápida

```
head -n 20 localstore.data | ./m.py
head -n 20 localstore.data | ./m.py | sort | ./r.py
```

Se pueden insertar prints con `DEBUG:` que Hadoop enviará a stderr.

## 7. Ejercicios completos

(ver archivo separado con todos los ejercicios del 1 al 15 incluidos, con código mapper.py y reducer.py desarrollado para cada uno)

```
# m_e1.py - Total facturado por ciudad origen
...
# r_e1.py
...

# m_e2.py - N° ventas por tipo de envío
...
# r_e2.py
...

# m_e3.py - Proveedor con más ventas
...
# r_e3.py
...

# m_e4.py - Factura total por año
...
# r_e4.py
...

# m_e5.py - Media de importe por proveedor y mes
...
# r_e5.py
...

# m_e6.py - Incremento porcentual anual por proveedor
...
```

```
# r_e6.py
...

# m_e7.py - Top-3 proveedores por gasto en un año
...
# r_e7.py
...

# m_e8.py - Venta máxima y mínima en un año
...
# r_e8.py
...

# m_e9.py - Densidad media de envíos tipo 1
...
# r_e9.py
...

# m_e10.py - Tipos de envío usados por proveedor
...
# r_e10.py
...

# m_e11.py - Ruta con más tráfico por año
...
# r_e11.py
...

# m_e12.py - Ciudades con más entrada/salida
...
# r_e12.py
...

# m_e13.py - Balance neto de ciudad
...
# r_e13.py
...

# m_e14.py - Coste medio de envío por proveedor
...
# r_e14.py
...


# m_e15.py - Histograma de importes
...
# r_e15.py
...
```

## 8. Trucos de rendimiento

Tema	Idea fuerza
Combiner	Reduce carga de red, ejecuta reduce parcial
Partitioner	Decide a qué reducer va cada clave
Fault-tolerance	Fallo parcial se recupera relanzando
Map Reduce vs Spark	Spark guarda en memoria, MR escribe a disco

## 9. Checklist pre-examen

- El mapper siempre emite con `\t`
- El reducer supone que la entrada viene ordenada
- ¿Gestión de errores? `continue`
- Prueba con `head -n 50 localstore.data`
- Para top-N: agrupa todo y luego filtra

 Con esta guía tienes explicaciones, patrones y una colección realista de ejercicios para MapReduce.

## Código completo de ejercicios 1 – 15

### Ejercicio 1 — Total facturado por ciudad origen

```
# m_e1.py - mapper
#!/usr/bin/python
import sys
for line in sys.stdin:
    cols = line.rstrip('
').split(' ')
    if len(cols) != 9:
        continue
    ciudad = cols[6].strip()
    try:
        importe = float(cols[8])
        print(f"{ciudad} {importe}")
    except:
        continue
```

```
# r_e1.py - reducer
#!/usr/bin/python
import sys
current = None
```

```

suma = 0.0
for line in sys.stdin:
    ciudad, valor = line.strip().split(' ')
    valor = float(valor)
    if ciudad == current:
        suma += valor
    else:
        if current is not None:
            print(f"{current} {suma:.2f}")
            current = ciudad
            suma = valor
if current is not None:
    print(f"{current} {suma:.2f}")

```

## Ejercicio 2 — Número de ventas por tipo de envío

```

# m_e2.py
#!/usr/bin/python
import sys
for l in sys.stdin:
    p = l.rstrip(' ')
    p).split(' ')
    if len(p) != 9:
        continue
    tipo = p[2].strip()
    print(f"{tipo} 1")

```

```

# r_e2.py
#!/usr/bin/python
import sys
cur = None
cnt = 0
for l in sys.stdin:
    k, v = l.strip().split(' ')
    if k == cur:
        cnt += int(v)
    else:
        if cur is not None:
            print(f"Tipo {cur} {cnt}")
            cur = k; cnt = int(v)
if cur is not None:
    print(f"Tipo {cur} {cnt}")

```

### Ejercicio 3 — Proveedor con mayor facturación total

```
# m_e3.py
#!/usr/bin/python
import sys
for line in sys.stdin:
    c = line.rstrip('
').split(' ')
    if len(c) != 9:
        continue
    prov = c[1].strip()
    try:
        imp = float(c[8])
        print(f"{prov} {imp}")
    except:
        continue
```

```
# r_e3.py
#!/usr/bin/python
import sys
from collections import defaultdict
suma = defaultdict(float)
for l in sys.stdin:
    prov, v = l.strip().split(' ')
    suma[prov] += float(v)
# buscar máximo
mejor, total = max(suma.items(), key=lambda x: x[1])
print(f"Proveedor con más ventas: {mejor} {total:.2f}")
```

---

### Ejercicio 4 — Factura total por año

*(ya incluido arriba, se mantiene igual)*

---

### Ejercicio 5 — Media de importe por proveedor + mes/año

*(ya incluido arriba, se mantiene igual)*

---

### Ejercicio 6 — Incremento porcentual anual por proveedor

*(ya incluido arriba, se mantiene igual)*

---

### Ejercicio 7 — Top-3 proveedores por gasto de envío en un año

*(ya incluido arriba, se mantiene igual)*

---

## Ejercicio 8 — Venta máxima y mínima en un año

(ya incluido arriba, se mantiene igual)

---

## Ejercicio 9 — Densidad media de envíos tipo 1 (peso / coste)

```
# m_e9.py
#!/usr/bin/python
import sys
for l in sys.stdin:
    c = l.rstrip('
').split(' ')
    if len(c) != 9: continue
    if c[2].strip() != '1':
        continue
    try:
        peso = float(c[3]); coste = float(c[4])
        dens = peso / coste if coste else 0
        print(f"dens    {dens}")
    except:
        continue
```

```
# r_e9.py
#!/usr/bin/python
import sys
suma = n = 0
for l in sys.stdin:
    _, v = l.strip().split(' ')
    suma += float(v); n += 1
if n:
    print(f"Densidad media (envíos tipo 1): {suma / n:.2f}")
```

---

## Ejercicio 10 — Tipos de envío utilizados por proveedor + mes/año

```
# m_e10.py
#!/usr/bin/python
import sys, datetime
for l in sys.stdin:
    c = l.rstrip('
').split(' ')
    if len(c) != 9: continue
    prov = c[1].strip(); tipo = c[2].strip()
    try:
        f = datetime.datetime.strptime(c[5], "%d/%m/%Y")
```



```

        key = f"{prov} {f.month:02d}/{f.year}"
        print(f"{key} {tipo}")
    except:
        continue

```

```

# r_e10.py
#!/usr/bin/python
import sys
cur = None; tipos = set()
for l in sys.stdin:
    k, t = l.strip().split(' ')
    if k == cur:
        tipos.add(t)
    else:
        if cur is not None:
            print(f"{cur}: {' '.join(sorted(tipos))}")
            cur = k; tipos = {t}
if cur is not None:
    print(f"{cur}: {' '.join(sorted(tipos))}")

```

## Ejercicio 11 — Ruta (origen→destino) con más tráfico por año

```

# m_e11.py
#!/usr/bin/python
import sys, datetime
for l in sys.stdin:
    c = l.rstrip(' ')
    c = c.split(' ')
    if len(c) != 9: continue
    try:
        año = datetime.datetime.strptime(c[5], "%d/%m/%Y").year
        ruta = f"{c[6]}→{c[7]}"
        print(f"{año} {ruta} 1")
    except:
        continue

```

```

# r_e11.py
#!/usr/bin/python
import sys
from collections import defaultdict
conteo = defaultdict(int)
for l in sys.stdin:
    a, r, n = l.strip().split(' ')
    conteo[(a, r)] += int(n)
por_año = defaultdict(list)
for (a, r), c in conteo.items():

```

```

        por_año[a].append((c, r))
for a in sorted(por_año):
    max_c, ruta = max(por_año[a])
    print(f"Año {a}: {ruta} - {max_c} envíos")

```

## Ejercicio 12 — Ciudades con mayor entrada + salida de envíos

```

# m_e12.py
#!/usr/bin/python
import sys
for l in sys.stdin:
    f = l.rstrip('
').split(' ')
    if len(f) != 9: continue
    origen = f[6].strip(); destino = f[7].strip()
    print(f"{origen}    1")
    print(f"{destino}  1")

```

```

# r_e12.py
#!/usr/bin/python
import sys
cur = None; tot = 0
for l in sys.stdin:
    k, v = l.strip().split(' ')
    if k == cur:
        tot += int(v)
    else:
        if cur is not None:
            print(f"{cur}    {tot}")
            cur = k; tot = int(v)
if cur is not None:
    print(f"{cur}    {tot}")

```

## Ejercicio 13 — Balance neto (salidas - entradas) de una ciudad

```

# m_e13.py
#!/usr/bin/python
import sys
for l in sys.stdin:
    p = l.rstrip('
').split(' ')
    if len(p) != 9: continue
    try:
        imp = float(p[8])
        print(f"{p[6]}  -{imp}") # sale
    except:
        pass

```

```

        print(f"{p[7]} {imp}") # entra
    except:
        continue

```

```

# r_e13.py
#!/usr/bin/python
import sys
cur = None; bal = 0.0
for l in sys.stdin:
    c, v = l.strip().split(' ')
    v = float(v)
    if c == cur:
        bal += v
    else:
        if cur is not None:
            print(f"{cur} {bal:.2f}")
            cur = c; bal = v
if cur is not None:
    print(f"{cur} {bal:.2f}")

```

## Ejercicio 14 — Coste medio de envío por proveedor

```

# m_e14.py
#!/usr/bin/python
import sys
for l in sys.stdin:
    c = l.rstrip('
').split(' ')
    if len(c) != 9: continue
    prov = c[1].strip()
    try:
        coste = float(c[4])
        print(f"{prov} {coste}")
    except:
        continue

```

```

# r_e14.py
#!/usr/bin/python
import sys
cur = None; s = n = 0
for l in sys.stdin:
    k, v = l.strip().split(' ')
    v = float(v)
    if k == cur:
        s += v; n += 1
    else:

```

```

        if cur is not None:
            print(f"{cur}    {s/n:.2f}")
        cur = k; s = v; n = 1
if cur is not None:
    print(f"{cur}    {s/n:.2f}")

```

## Ejercicio 15 — Histograma de importes (rangos 100 €)

```

# m_e15.py
#!/usr/bin/python
import sys
BIN = 100 # anchura del bucket
for l in sys.stdin:
    p = l.rstrip('
').split(' ')
    if len(p) != 9: continue
    try:
        imp = float(p[8])
        bucket = int(imp // BIN) * BIN
        label = f"{bucket}-{bucket+BIN}"
        print(f"{label} 1")
    except:
        continue

```

```

# r_e15.py
#!/usr/bin/python
import sys
cur = None; cnt = 0
for l in sys.stdin:
    b, v = l.strip().split(' ')
    if b == cur:
        cnt += int(v)
    else:
        if cur is not None:
            print(f"{cur} € {cnt}")
        cur = b; cnt = int(v)
if cur is not None:
    print(f"{cur} € {cnt}")

```