Cálculo de la media mediante map-reduce con post-procesamiento

El porqué del post-procesamiento. Conceptos básicos

La operación de *sumar* los pares es asociativa:

```
[2, 3, 4] -> [(2, 1), (3, 1), (4, 1)] -> [(5, 2), (4, 1)] -> (9, 3)
[2, 3, 4] -> [(2, 1), (3, 1), (4, 1)] -> [(2, 1), (7, 2)] -> (9, 3)
```

Pueden realizarse sumas en un orden y agrupamiento arbitrarios, y concretamente en el que proporcionen los distintos procesadores de forma distribuida. Por lo tanto, se puede aplicar la operación `reduce` con seguridad.

Deseamos ahora *calcular la media* de una lista de números. Por ejemplo, la siguiente:

$$[2, 3, 4]$$

Observamos que esta operación no es asociativa, y que la asociación en cualquier orden no conduce a la media verdadera:

$$\frac{\frac{2+3}{2}+4}{2} \neq \frac{2+\frac{3+4}{2}}{2} \neq \frac{2+3+4}{3}$$

Así pues, la simple aplicación de un algoritmo de map-reduce no es tan simple, pues el resultado dependerá de en qué orden vayan proporcionándose los resultados parciales, generados de forma distribuida.

Sin embargo, es posible aplicar otro enfoque: calcular las sumas de los valores y del número de valores:

```
[2, 3, 4] -> [(2, 1), (3, 1), (4, 1)] -> (9, 3)
```

Evidentemente, este algoritmo no calcula la media, sino un par de valores. Por lo tanto, requiere un paso adicional, posteriormente: dividir ese par de valores.

Esta operación que se debe hacer **tras el cálculo intensivo** es la que nos ayuda a ilustrar cómo aplicar algún tipo de postprocesamiento.

Nuestro ejemplo concreto

Tenemos una lista de colores: `{Azul, Blanco, Verde, Amarillo}` . Cada color tiene asociado un valor entero:

```
Valores = {"Azul": 3.0, "Blanco": 5.0, "Verde": 15.0, "Amarillo": 10.0}
```

Elegimos un color, con igual probabilidad, entre la lista `Colores` , y para cada color, generamos un número real, extraído uniformemente en el intervalo del valor del color, ± 1. Por ejemplo,m si el color elegido es el azul, el valor será un número real del intervalo $[3-1, 3+1]$.

In [1]:

```
import random

Valores = {"Azul": 3.0, "Blanco": 5.0, "Verde": 15.0, "Amarillo": 10.0}
Colores = list(Valores.keys())

def v_a(color):
    valor = random.random() * 2 + Valores[color] - 1
    return round(valor, 2)

def generar_datos(n, archivo):
    with open(archivo, "w") as f:
        for _ in range(n):
            [color] = random.choices(Colores)
            x = v_a(color)
            f.write(color + " " + str(x) + "\n")
```

Para calcular las medias de cada color, generamos una muestra, en un archivo de texto.

In [2]:

```
generar_datos(20, "datos.txt")

! type "datos.txt"
```

```
Amarillo 9.48
Azul 2.18
Blanco 4.71
Verde 14.4
Azul 3.32
Verde 15.27
Amarillo 10.72
Verde 15.96
Azul 2.19
Verde 14.52
Amarillo 10.04
Blanco 5.93
Azul 2.41
Blanco 4.61
Amarillo 9.1
Blanco 4.35
Blanco 5.52
Azul 3.14
Azul 3.3
Verde 14.77
```

Generamos ahora una muestra más reealista:

In [3]:

```
generar_datos(1000, "datos.txt")
```

In [4]:

```
# Veamos el contenido de esta carpeta tras la generación de este archivo:

! dir
```

```
 El volumen de la unidad C es Windows
 El n£mero de serie del volumen es: 22D6-2907

 Directorio de C:\Users\Cristobal\Jupyter\Python E1 - librer¡as - an lisis d
e datos\E7 - map-reduce\caso_3 - media - postprocesamiento

16/10/2022  12:27    <DIR>          .
15/10/2022  17:15    <DIR>          ..
11/10/2022  17:28    <DIR>          .ipynb_checkpoints
16/10/2022  12:30            13.036 datos.txt
05/03/2021  16:19             3.944 E7 - ej3 - medias y postprocesamiento.ra
r
04/03/2021  10:12               521 generar_archivo.py
16/10/2022  12:30               858 medias.py
16/10/2022  12:27             8.793 media_post.ipynb
16/10/2022  12:09           176.492 media_post.pdf
               6 archivos        203.644 bytes
               3 dirs  218.703.601.664 bytes libres
```

Problema: cálculo de la media de cada color

Aquí es donde se plantea resolver el cálculo de la media por color, con un programa map-reduce.

Se ha planteado un programa para ser ejecutado desde la consola. Veámoslo.

In [5]:

```
! python medias.py datos.txt -q
```

```
Amarillo - 9.98
Azul - 3.02
Blanco - 4.93
Verde - 15.06

No configs specified for inline runner
```

In [6]:

```
! type medias.py
```

```
ï»¿
import sys
from mrjob.job import MRJob

def suma_doble(pares):
    """Ej. [(1, 10), (2, 20), (3, 30)]  --> (6, 60)"""
    a, b = 0, 0
    for x, y in pares:
        a, b = a + x, b + y
    return a, b

class MRSumaTotales(MRJob):

    def mapper(self, _, linea):
        [color, x] = linea.split()
        yield color, (float(x), 1)

    def reducer(self, key, values):
        yield key, suma_doble(values)

if __name__ == '__main__':

    archivo_datos = sys.argv[1]
    trabajo = MRSumaTotales(args=[archivo_datos])
    with trabajo.make_runner() as runner:
        runner.run()
        for key, value in trabajo.parse_output(runner.cat_output()):
            media = value[0] / value[1]
            media_str = str(round(media, 2))
            print(key + " - " + media_str)
```

Nota final

Se trata de unos apuntes breves que intentan ser sobre todo útiles y claros. Obviamente, se han omitido los comentarios del programa de `map-reduce` por brevedad, esperando que sea lo bastante claro así.