

Trabajo Practico N° 6

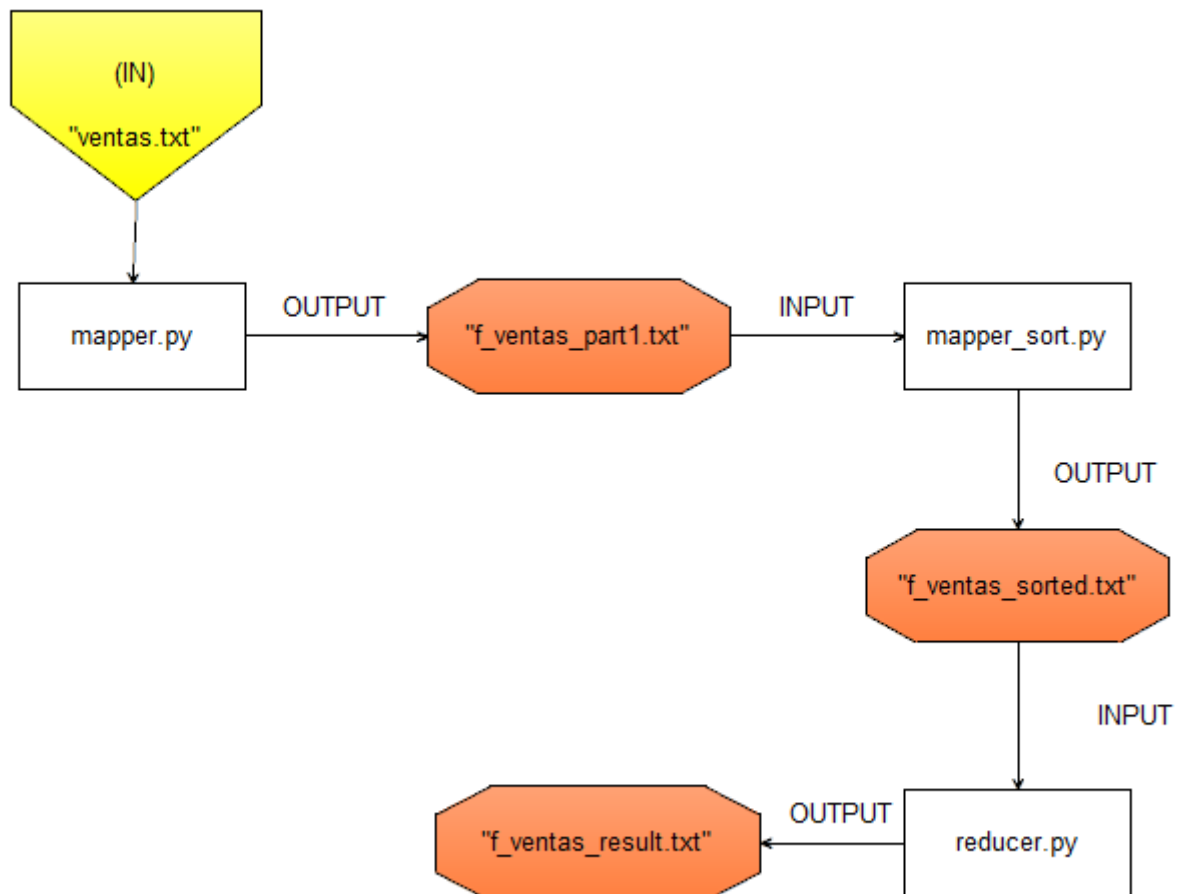
Nociones de Frameworks de Procesamiento Masivo

Genere un esquema bajo el paradigma MapReduce para resolver las siguientes consignas:

a) Produzca un mapper y un reducer para responder a cuál es el bonus obtenido por cada vendedor siendo que cada vendedor obtiene el 3% del total del dinero vendido.

$X * 0.03$ = es el 3% del total vendido

agrego un diagrama explicativo de los algoritmos para la resolucion del punto.



Los script ordenados son como sigue:

- mapper.py : en esta parte creo la estructura {clave valor}, falta ordenar..
- mapper_sort.py : aqui ordeno por id_vendedor, quedandome como resultado una estructura que falta reducir.

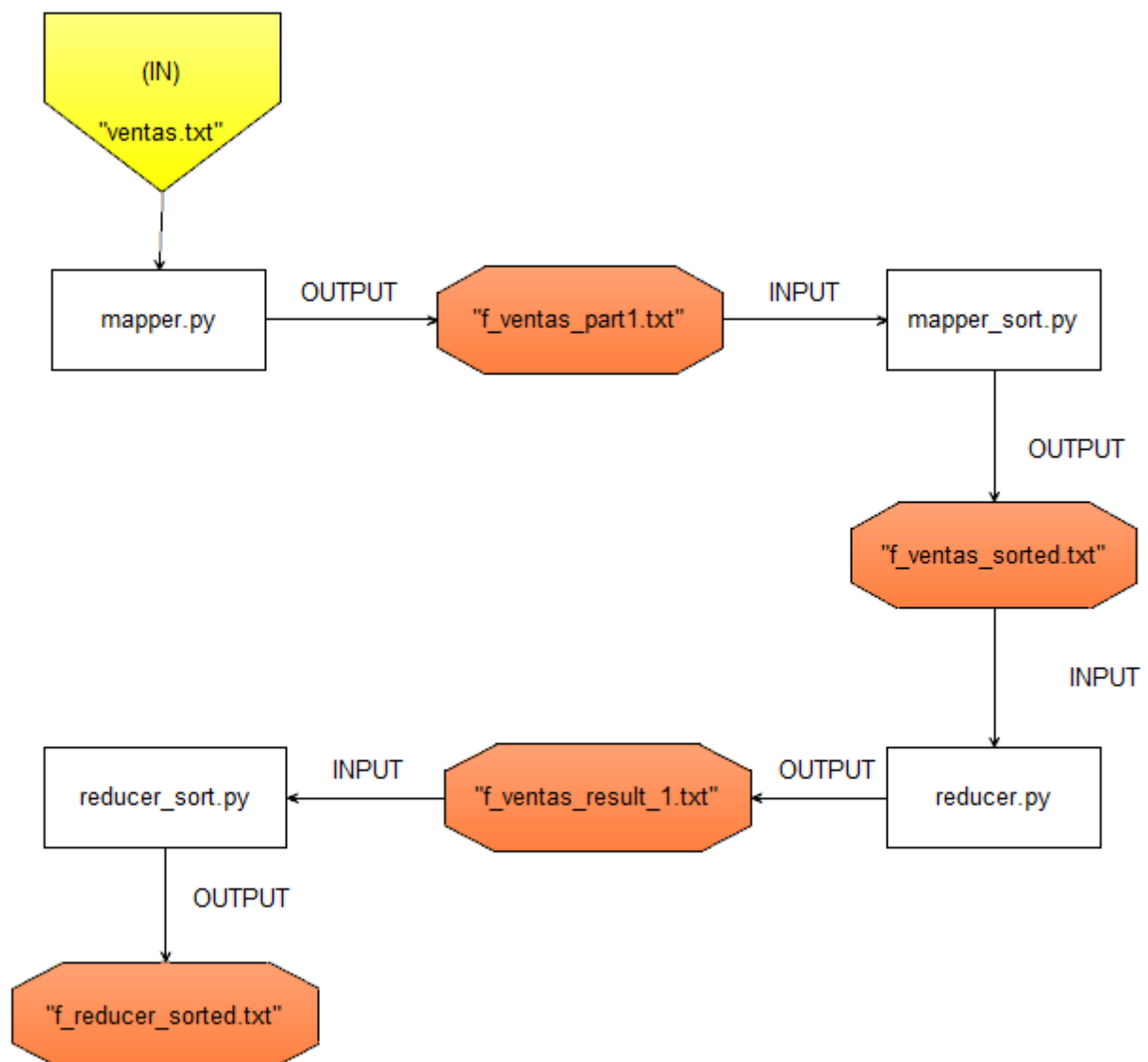
- reducer.py : voy comparando linea a linea y cuando el "id_vendedor" cambia renuevo el contador y guardo en un archivo de salida "f_ventas_result.py".

Enlace a la carpeta con los script y archivos resultantes [Punto 1](#)

Agregue un script para correr todos los "script.py" juntos y en orden.

b) Produzca un mapper y un reducer para obtener la cantidad de productos vendidos por cada vendedor, agrupado por coordinador.

Agrego un diagrama explicativo del algoritmo.



Los pasos son como siguen:

- mapper.py : en esta parte armo el esquema {id_coordinador, id_vendedor, cant_prod_vendidos}, aun no sumo los totales por vendedor. Como salida tengo una lista, con la estructura [id_coordinador, id_vendedor, cant_prod_vendidos]
- mapper_sort.py : aqui ordeno por vendedor quedandome de esta forma

```
e > punto-2 > ≡ f_ventas_sorted.txt
1 6620 100 12
2 6620 100 38
3 6620 100 42
4 6620 100 46
5 6620 100 28
6 6620 100 9
7 6620 100 3
8 6620 100 33
9 6620 100 10
10 6620 100 45
11 6620 100 23
12 6620 100 19
13 8501 101 38
```

en este algoritmo la parte mas importante es la de ordenamiento..

```
def convertir_id(line):
    line = line.strip()
    linea_split = []
    linea_split = line.split("\t")
    # PARTE IMPORTANTE DEL SORT, SELECCIONO LA COLUMNA POR LA CUAL
    # QUIERO ORDENAR, EN ESTE CASO COLUMNA ID_VENDEDOR
    # PARA DESPUES HACER EL ACUMULADOR DE CANT_VENTAS_REALIZADAS
    return int(linea_split[1])
```

Retornando la columna por la cual quiero ordenar. Luego el ciclo for hace el resto..

- reducer.py: En esta parte sumo los subtotales de productos vendidos por vendedor quedando el resultado como sigue

```
code > punto-2 > ≡ f_ventas_result_1.txt
1 6620 100 308
2 8501 101 386
3 19999 102 206
4 14626 103 385
5 4527 105 131
6 18381 107 281
7 9227 108 162
8 11180 109 262
9 599 110 234
10 21016 112 199
11 23788 115 165
12 7651 116 109
13 21834 118 192
14 13370 122 107
15 2154 124 95
16 8152 126 373
```

aca tengo los vendedores con los totales vendidos. Ahora me falta hacer un orden por vendedor.

- reducer_sort.py: aplico el mismo paso de ordenamiento realizado en el "mapper_sorted" pero para ordenar por coordinador

```
code > punto-2 > f_reducer_sorted.txt
1   ID_COORDINADOR  ID_VENDEDOR  TOTAL_VENDIDO
2   103 1630        213
3   103 6896        110
4   103 9549        190
5   103 10770       292
6   103 24620       165
7   103 26854       389
8   103 29210       212
9   131 487 286
10  131 7231        147
11  131 10163       256
12  131 10219       135
13  131 12947       106
14  131 14713       179
15  131 16361       133
16  131 16814       183
17  131 17119       115
```

Enlace a la carpeta con los script y archivos resultantes [Punto 2](#)

Agregue un script para correr todos los "script.py" juntos y en orden.

Apache Spark con PySpark: Resuelva el ejercicio anterior con PySpark.

La parte mas importante de la consigna es como sigue

```
RDD_ventas = sc.textFile("/content/ventas.txt").\
    map(lambda line: line.split("\t")).\
    map(lambda d: (int(d[0]), int(d[2]))).\
    reduceByKey(lambda x, y: x + y)

RDD_ventas.sortByKey().collect()
```

De lo que se obtiene todos los vendedores ordenados y con el total de productos vendidos.

Me queda por concluir, poder agrupar los vendedores por coordinador. Aun estoy trabajando en ello.

Adjunto el enlace al [Colab](#)