ALUMNO: VICTOR JOSE QUISPE HUARCAYA

CODIGO: N00311875

DATASCIENCE

DATASCIENCE CON APACHE SPARK

¿Qué es APACHE SPARK?

SPARK es un framework de procesamiento de datos distribuido y de código abierto diseñado para analizar grandes cantidades de volúmenes de datos de manera rápida y eficiente.

¿Cómo funciona?

- Almacena datos en memoria para acelerar las operaciones, y se puede integrar con Hadoop, siendo compatible con el sistema de archivos HDFS
- Los datos se dividen en RDD`s, que se distribuyen en el cluster
- Las transformaciones o acciones se realizan en los RDDs para generar nuevos datos. Estas transformaciones se registran en un DAG optimizando el flujo de trabajo y permitiendo la reconstrucción en caso de fallos.

Transformaciones

Las transformaciones son operaciones que toman un RDD y producen un nuevo RDD como resultado.

- **map(función)**: Aplica una función a cada elemento del RDD y devuelve un nuevo RDD con los resultados.
- **filter(función)**: Filtra los elementos del RDD que cumplen con una condición específica.
- **flatMap(función)**: Similar a map(), pero puede devolver múltiples elementos para cada entrada.
- **reduceByKey(función)**: Agrupa los valores por clave y luego aplica una función de reducción a los valores de cada grupo

Acciones

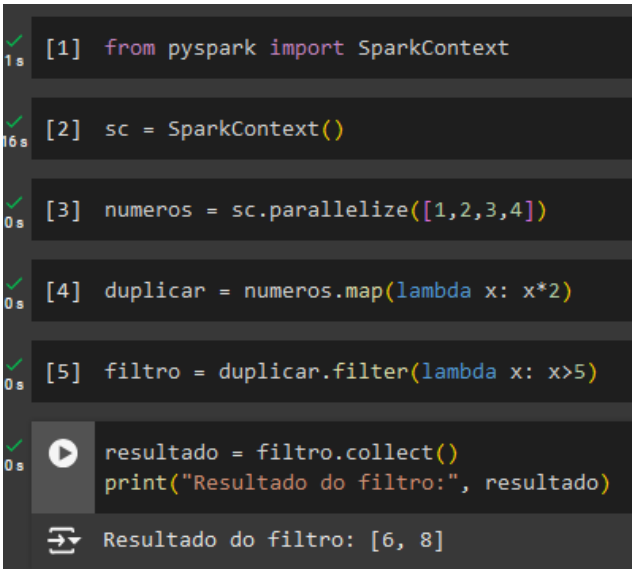
Las acciones son operaciones que desencadenan la ejecución de las transformaciones acumuladas en el DAG y devuelven un resultado.

- **collect()**: Recupera todos los elementos del RDD en la máquina local como una lista.
- **count()**: Cuenta el número de elementos en el RDD.
- **take(n)**: Devuelve los primeros n elementos del RDD.

Ejemplos de uso:

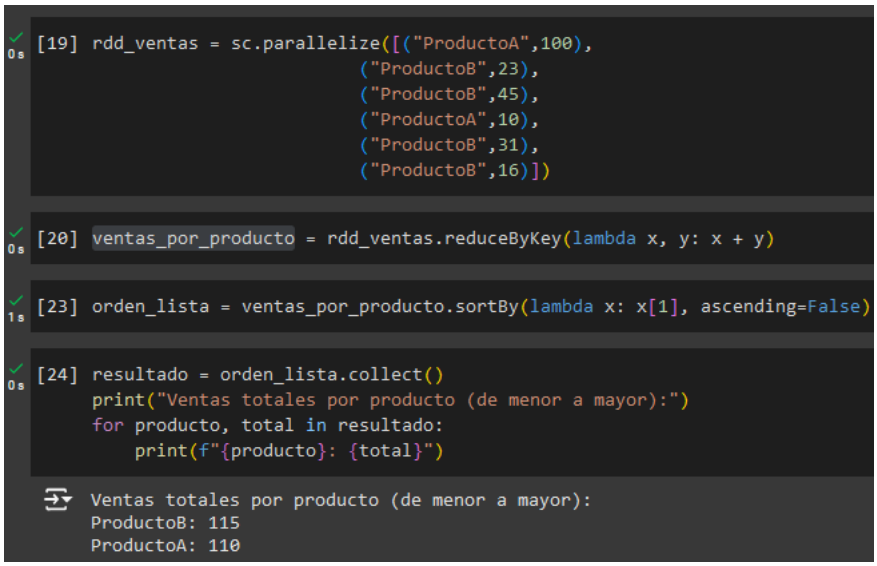
CASO 1

- Transformación map: Se utiliza para multiplicar cada número
- Transformación filter: Filtrar los números que son mayores a 5
- Acción collect: Recolecta el resultado desde el RDD



CASO 2

- reduceByKey: Agrupa los datos y suma las cantidades
- sortBy: Ordena el resultado del RDD



CASO 3

- Se ingreso una base de datos de ingresos
- Se conto los accesos por día
- Se separo el número de accesos por sedes.

```
[48] registros_rdd = sc.parallelize ([
         ("2024-11-10", "UPN-CHORRILLOS", "user1"),
         ("2024-11-10", "UPN-CHORRILLOS", "user2"),
         ("2024-11-10", "UPN-CHORRILLOS", "user3"),
         ("2024-11-11", "UPN-BREÑA", "user4"),
         ("2024-11-11", "UPN-CHORRILLOS", "user5"),
         ("2024-11-11", "UPN-BREÑA", "user1"),
         ("2024-11-12", "UPN-CHORRILLOS", "user6"),
         ("2024-11-12", "UPN-CHORRILLOS", "user2"),
         ("2024-11-12", "UPN-BREÑA", "user2"),
         ("2024-11-12", "UPN-CHORRILLOS", "user4"),
         ("2024-11-12", "UPN-CHORRILLOS", "user1"),
     ])

[49] contador_accesos = registros_rdd.map(lambda x: (x[0], 1)).reduceByKey(lambda x, y: x + y)

[50] resultado_dia = contador_accesos.collect()
     print("Accesos por día:")
     for fecha, total in resultado_dia:
         print(f"{fecha}: {total}")

     Accesos por día:
     2024-11-10: 3
     2024-11-11: 3
     2024-11-12: 5

[51] sedes = registros_rdd.map(lambda x: (x[1], 1)).reduceByKey(lambda x, y: x + y)

[52] orden_sede = sedes.sortBy(lambda x: x[1], ascending=False)

[55] resultado_sede = orden_sede.collect()
     print("Ingresos por sede (de mayor a menor):")
     for sede, total in resultado_sede:
         print(f"{sede}: {total}")

     Ingresos por sede (de mayor a menor):
     UPN-CHORRILLOS: 8
     UPN-BREÑA: 3
```

CASO 4

- Usaremos un archivo Excel para analizar su registro de ventas
- Tener en cuenta que todo se esta trabajando en un mismo archivo collab. Por lo que, al final se va a dar por finalizada la sesion de **SPARKCONTEXT**.
- Tener en cuenta el uso del 'Date Time'
- Se agrupara por categoría y como resultado se dará la suma total de cada una
- Se agrupara la cantidad total por producto
- Se mostrara el total de ventas totales por mes

```
[56] import pandas as pd

     df = pd.read_excel("empresa_x2.xlsx")
```

```
[57] datos = [tuple(x) for x in df.to_records(index=False)]
```

```
[58] rdd_ventas = sc.parallelize(datos)
```

```
[70] df
```

	Fecha	Producto o Servicio	Categoría	Cantidad Vendida	Precio Unitario	Total Venta
0	2024-01-01	Laptop Pro	Hardware	35	1079.96	37798.60
1	2024-01-01	Tablet Lite	Hardware	48	295.64	14190.72
2	2024-01-01	Software CRM	Software	47	1055.52	49609.44
3	2024-01-01	Soporte Técnico Mensual	Suscripción	5	739.12	3695.60
4	2024-02-01	Laptop Pro	Hardware	21	191.81	4028.01
5	2024-02-01	Tablet Lite	Hardware	18	357.72	6438.96
6	2024-02-01	Software CRM	Software	37	213.23	7889.51
7	2024-02-01	Soporte Técnico Mensual	Suscripción	34	950.94	32331.96
8	2024-03-01	Laptop Pro	Hardware	36	908.08	32690.88
9	2024-03-01	Tablet Lite	Hardware	42	882.44	37062.48
10	2024-03-01	Software CRM	Software	32	663.07	21218.24
11	2024-03-01	Soporte Técnico Mensual	Suscripción	10	863.86	8638.60
12	2024-03-01	Laptop Pro	Hardware	30	1079.96	32398.80
13	2024-03-01	Tablet Lite	Hardware	50	295.64	14782.00
14	2024-03-01	Software ERP	Software	40	1255.32	50212.80
15	2024-03-01	Soporte Técnico Anual	Suscripción	3	739.12	2217.36

```
[59] # Sumar las ventas totales por categoría
     ventas_por_categoria = rdd_ventas.map(lambda x: (x[2], x[5])).reduceByKey(lambda x, y: x + y)

     resultado_categoria = ventas_por_categoria.collect()
     print("Ventas totales por categoría:")
     for categoria, total in resultado_categoria:
         print(f"{categoria}: {total}")
```

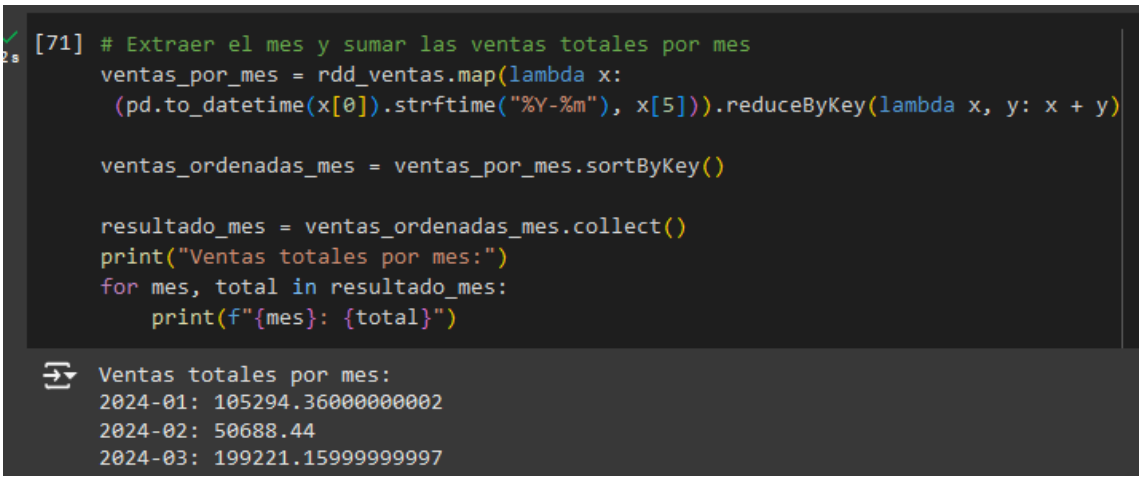
```
Ventas totales por categoría:
Suscripción: 46883.52
Hardware: 179390.45
Software: 128929.99000000002
```

```
[64] # Sumar la cantidad vendida por producto
     cantidad_por_producto = rdd_ventas.map(lambda x: (x[1], x[3])).reduceByKey(lambda x, y: x + y)

     orden_empresa = cantidad_por_producto.sortBy(lambda x: x[1], ascending=False)

     resultado_producto = orden_empresa.collect()
     print("Producto más vendido por cantidad:")
     for producto, cantidad in resultado_producto:
         print(f"{producto}: {cantidad}")
```

```
Producto más vendido por cantidad:
Tablet Lite: 158
Laptop Pro: 122
Software CRM: 116
Soporte Técnico Mensual: 49
Software ERP: 40
Soporte Técnico Anual: 3
```



```
[71] # Extraer el mes y sumar las ventas totales por mes
     ventas_por_mes = rdd_ventas.map(lambda x:
      (pd.to_datetime(x[0]).strftime("%Y-%m"), x[5])).reduceByKey(lambda x, y: x + y)

     ventas_ordenadas_mes = ventas_por_mes.sortByKey()

     resultado_mes = ventas_ordenadas_mes.collect()
     print("Ventas totales por mes:")
     for mes, total in resultado_mes:
         print(f"{mes}: {total}")

Ventas totales por mes:
2024-01: 105294.36000000002
2024-02: 50688.44
2024-03: 199221.15999999997
```

Conclusión

Spark permite a las organizaciones realizar análisis complejos de manera eficiente y es un recurso clave en la tecnología de Big Data actual debido a su capacidad de procesamiento en memoria y su facilidad de uso la hacen una opción superior para muchos casos de uso.