# ejercicio_3_miguelperezcaro

April 18, 2021

# 1 PEC PROCESAMIENTO DE STREAMS E INGESTA DE DATOS

## 1.1 MIGUEL PÉREZ CARO

### 1.1.1 Ejercicio 3: Spark

Una empresa que controla el movimiento de vehículos ha implementado un sistema con diez puntos de control en una ciudad. Cada vez que un vehículo pasa por uno de los puntos se recoge su matrícula y en tiempo real se envía la información de la matrícula junto al punto por donde pasó ese vehículo con la siguiente estructura, MATRICULA PUNTO. ąOJO! La información está separada por un espacio en blanco, teneis que hacer las simulaciones con este formato. Ejs. 12345BTS PUNTO1 o 6789FYZ PUNTO2 Nos solicita que hagamos el siguiente control en tiempo real con Spark:

En primer lugar se importan las librerías necesarias

```
In [1]: import findspark
        findspark.init("/home/ubuntu/bigdata/apps/spark")

        from pyspark import SparkContext
        from pyspark.streaming import StreamingContext
```

**Apartado A** Saber en tiempo real cuantas veces pasa un vehículo por cada punto del sistema desde el día que se pone en marcha el mecanismo.

Para la realización de este ejercicio se comienza creando el contexto necesario.

```
In [2]: sc.stop()
        sc = SparkContext(appName="Ejercicio3Apartado1")
        ssc = StreamingContext(sc, 10)
```

A continuación se genera una función para calcular el añadido

```
In [3]: def update_func(new_val, last_sum):
            """
            new_val: nuevo valor que se procesa
            last_sum: antiguo valor que ya ha sido procesado
                      o 0 si no había valor procesado anteriormente
            """
            return sum(new_val) + (last_sum or 0)
```

Se genera un directorio donde se van guardando los resultados a modo de checkpoint y al que se podría recurrir en caso de problema.

```
In [4]: checkpointDir = "file:///home/ubuntu/bigdata/apps/spark/checkpoint"
        ssc.checkpoint(checkpointDir)
```

Se recoge la información simulada a través de un socket y se preprocesa. Hay que tener en cuenta que el sistema se ha diseñado considerando que cada matrícula con el punto de localización se añade de forma individual, independientemente de que la diferencia entre los coches pasando por un mismo punto sea de milisegundos. También hay que destacar que el formato en el que se recibe la información es el siguiente:

- MATRÍCULA PUNTO tal que 1234ABC PUNTO1
- La información devuelta tiene que ser con la información en formato tupla tal que el primer elemento ha de ser 1234ABC-PUNTO1, y el segundo el número de veces que dicho coche ha pasado por ese punto.

```
In [5]: # Recogida de información
        lines = ssc.socketTextStream("localhost", 10001)

        # Transformación de formato
        words = lines.map(lambda line: line.replace(" ", '-'))
        # Generar tupla y agregar a lo calculado
        counts = words.map(lambda word: (word, 1)).updateStateByKey(update_func)

        counts.pprint()
```

La simulación de información se hace a través de la consola ejecutando el comando nc -lk 10001. A continuación se comienza el proceso:

```
In [6]: ssc.start()

-------------------------------------------
Time: 2021-04-16 17:04:00
-------------------------------------------

-------------------------------------------
Time: 2021-04-16 17:04:10
-------------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO1', 2)

-------------------------------------------
Time: 2021-04-16 17:04:20
-------------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO4', 1)
('1234ABC-PUNTO1', 4)
('1234ABC-PUNTO3', 1)
```

```
-----------------------------------------
Time: 2021-04-16 17:04:30
-----------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO4', 1)
('1234ABC-PUNTO1', 4)
('1234ABC-PUNTO3', 1)
('1234ABC-PUNTO5', 1)

-----------------------------------------
Time: 2021-04-16 17:04:40
-----------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO4', 1)
('5678AAA-PUNTO1', 1)
('1234ABC-PUNTO1', 5)
('1234ABC-PUNTO3', 1)
('1234ABC-PUNTO5', 1)

-----------------------------------------
Time: 2021-04-16 17:04:50
-----------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO4', 1)
('5678AAA-PUNTO1', 1)
('1789QWE-PUNTO4', 1)
('1234ABC-PUNTO1', 5)
('1234ABC-PUNTO3', 1)
('1234ABC-PUNTO5', 1)

-----------------------------------------
Time: 2021-04-16 17:05:00
-----------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO4', 1)
('5678AAA-PUNTO1', 1)
('1789QWE-PUNTO4', 1)
('1234ABC-PUNTO1', 5)
('1234ABC-PUNTO3', 1)
('1234ABC-PUNTO5', 1)
('2389ASD-PUNTO9', 1)

-----------------------------------------
Time: 2021-04-16 17:05:10
-----------------------------------------
('1234ABC-PUNTO2', 3)
('1234ABC-PUNTO4', 2)
```

```
('5678AAA-PUNTO1', 1)
('1789QWE-PUNTO4', 1)
('1234ABC-PUNTO1', 5)
('1234ABC-PUNTO3', 2)
('1234ABC-PUNTO5', 1)
('2389ASD-PUNTO9', 1)

-------------------------------------------
Time: 2021-04-16 17:05:20
-------------------------------------------
('1234ABC-PUNTO2', 3)
('1234ABC-PUNTO4', 2)
('5678AAA-PUNTO1', 1)
('1789QWE-PUNTO4', 1)
('1234ABC-PUNTO1', 5)
('1234ABC-PUNTO3', 2)
('1234ABC-PUNTO5', 1)
('2389ASD-PUNTO9', 1)

-------------------------------------------
Time: 2021-04-16 17:05:30
-------------------------------------------
('1234ABC-PUNTO2', 3)
('1234ABC-PUNTO4', 2)
('5678AAA-PUNTO1', 1)
('1789QWE-PUNTO4', 1)
('1234ABC-PUNTO1', 5)
('1234ABC-PUNTO3', 2)
('1234ABC-PUNTO5', 2)
('2389ASD-PUNTO9', 1)

-------------------------------------------
Time: 2021-04-16 17:05:40
-------------------------------------------
('1234ABC-PUNTO2', 3)
('1234ABC-PUNTO4', 2)
('5678AAA-PUNTO1', 2)
('1789QWE-PUNTO4', 1)
('1234ABC-PUNTO1', 5)
('1234ABC-PUNTO3', 2)
('1234ABC-PUNTO5', 2)
('2389ASD-PUNTO9', 2)
```

En el resultado de la celda anterior se ve como se van procesando las matrículas con el punto de localización, y como se van añadiendo según llegan la misma combinación de matrícula y punto.

Por último, se paran los procesos.

```
In [7]: ssc.stop()
        sc.stop()
```

**Apartado B** Saber en tiempo real cuántas veces pasa un vehículo por un determinado punto, pero solo teniendo en cuenta los últimos 7 días de información.

Este apartado es idéntico al anterior pero hay que añadir una ventana de 7 días, de forma que cuando hayan pasado esos 7 días, se borren los datos. Se comienza nuevamente generando el contexto.

```
In [8]: sc.stop()
        sc = SparkContext(appName="Ejercicio3Apartado2")
        ssc = StreamingContext(sc, 10)
```

Se crea otro directorio checkpoint.

```
In [9]: checkpointDir2 = "file:///home/ubuntu/bigdata/apps/spark/checkpoint2"
        ssc.checkpoint(checkpointDir)
```

Se realiza una transformación similar a la anterior, pero en este caso hay que utilizar la función reduceByKeyAndWindow que nos permite ir agregando las combinaciones de matrícula y punto de localización, y a su vez establecer una ventana a partir de la cual se vayan restando dichos valores. Como la ventana es de 7 días, no se va a apreciar visualmente como se eliminarían los puntos una vez hubiera pasado dicho espacio de tiempo.

```
In [10]: lines = ssc.socketTextStream("localhost", 10002)
         words = lines.map(lambda line: line.replace(" ", '-'))
         # Se añade la función reduceByKeyAndWindow con ventana de 7 días que serían 604800s.
         counts = words.map(lambda word: (word, 1)).reduceByKeyAndWindow(lambda x, y: x+y,
                                                                         lambda x, y: x-y,
                                                                         604800,
                                                                         10)

         counts.pprint()

In [11]: ssc.start()

-------------------------------------------
Time: 2021-04-16 17:36:30
-------------------------------------------

-------------------------------------------
Time: 2021-04-16 17:36:40
-------------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO1', 2)
('1234ABC-PUNTO3', 1)

-------------------------------------------
Time: 2021-04-16 17:36:50
-------------------------------------------
```

```
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO4', 2)
('1234ABC-PUNTO6', 1)
('1234ABC-PUNTO1', 2)
('1234ABC-PUNTO3', 2)
('1234ABC-PUNTO5', 1)

-------------------------------------------
Time: 2021-04-16 17:37:00
-------------------------------------------
('1234ABC-PUNTO2', 1)
('1234ABC-PUNTO4', 2)
('1234ABC-PUNTO6', 1)
('1234ABC-PUNTO1', 2)
('1234ABC-PUNTO3', 4)
('1234ABC-PUNTO5', 3)
('1234ABC-PUNTO7', 1)

-------------------------------------------
Time: 2021-04-16 17:37:10
-------------------------------------------
('1234ABC-PUNTO2', 2)
('1234ABC-PUNTO4', 2)
('1234ABC-PUNTO6', 1)
('1234ABC-PUNTO1', 4)
('1234ABC-PUNTO3', 4)
('1234ABC-PUNTO5', 4)
('1234ABC-PUNTO7', 1)

-------------------------------------------
Time: 2021-04-16 17:37:20
-------------------------------------------
('1234ABC-PUNTO2', 2)
('1234ABC-PUNTO4', 2)
('1234ABC-PUNTO6', 1)
('1234ABC-PUNTO1', 4)
('1234ABC-PUNTO3', 4)
('1234ABC-PUNTO5', 4)
('6789AAA-PUNTO1', 1)
('1234ABC-PUNTO7', 1)
```

```
In [12]: ssc.stop()
         sc.stop()
```

Se aprecia un resultado final idéntico al que se hubiera logrado con la función utilizada en el apartado A, pero en este caso se ha añadido la ventana de 7 días, de forma que esos datos se eliminarían una vez pasado dicho espacio de tiempo.